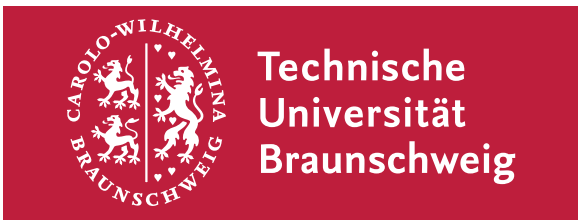
Detecting and Mitigating Malicious Behavior in Vehicular DTNs



Von der Carl-Friedrich-Gauß-Fakultät
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades einer
Doktoringenieurin (Dr.-Ing.)

genehmigte Dissertation

von

Yinghui Guo
geboren am 29.10.1985
in Jilin

Eingereicht am:	11. 08. 2014
Disputation am:	02. 12. 2014
1. Referent:	Prof. Dr. Lars Wolf
2. Referent:	Prof. Dr. Frank Kargl

2014

Abstract

Delay- and Disruption-Tolerant Networks (DTNs) are a suitable technology for many applications when the network suffers from intermittent connections and significant delays. In current vehicular networks, due to the high mobility of vehicles, the connectivity in vehicular networks can be highly unstable, links may change or break soon after they have been established and the network topology varies significantly depending on time and location. When the density of networked vehicles is low, connectivity is intermittent and with only a few transmission opportunities. This makes forwarding packets very difficult. For the next years, until a high penetration of networked vehicles is realized, delay-tolerant methods are a necessity in vehicular networks, leading to Vehicular DTNs (VDTNs). By implementing a store-carry-forward paradigm, VDTNs can make sure that even under difficult conditions, the network can be used by applications. However, we cannot assume that all vehicles are altruistic in VDTNs. Attackers can penetrate the communication systems of vehicles trying their best to destroy the network. Especially if multiple attackers collude to disrupt the network, the characteristics of VDTNs, without continuous connectivity, make most traditional strategies of detecting attackers infeasible. Additionally, selfish nodes may be reluctant to cooperate considering their profit, and due to hard- or software errors some vehicles cannot send or forward data. Hence, efficient mechanisms to detect malicious nodes in VDTNs are imperative. In this thesis, two classes of Misbehavior Detection Systems (MDSs) are proposed to defend VDTNs against malicious nodes. Both MDSs use encounter records (ERs) as proof to document nodes' behavior during previous contacts. By collecting and securely exchanging ERs, depending on different strategies in different classes of MDSs, a reputation system is built in order to punish bad behavior while encouraging cooperative behavior in the network. With independently operating nodes and asynchronous exchange of observations through ERs, both systems are very well suited for VDTNs, where there will be no continuous, ubiquitous network in the foreseeable future. By evaluating our methods through extensive simulations using different DTN routing protocols and different realistic scenarios, we find that both MDS classes are able to efficiently protect the system with low overhead and prevent malicious nodes from further disrupting the network.

Kurzfassung

In Netzwerken mit zeitweisen Unterbrechungen oder langen Verzögerungen sind Delay- and Disruption-Tolerant Networks (DTNs) eine geeignete Technologie für viele Anwendungen. Die Konnektivität in Fahrzeugnetzen ist bedingt durch die hohe Mobilität und die geringe Verbreitung von netzwerkfähigen Fahrzeugen oft instabil. Bis zur flächendeckenden Verbreitung von netzwerkfähigen Fahrzeugen ist es daher zwingend notwendig auf Methoden des Delay Tolerant Networking zurückzugreifen um die bestmögliche Kommunikation zu gewährleisten. In diesem Zusammenhang wird von Vehicular Delay Tolerant Networks (VDTNs) gesprochen. Durch das Store-Carry-Forward-Prinzip kann ein VDTN Kommunikation für Anwendungen ermöglichen. Allerdings ist davon auszugehen, dass sich nicht alle Fahrzeuge altruistisch verhalten: Angreifer können Fahrzeuge übernehmen und das Netzwerk attackieren oder Knoten sind aus egoistischen Motiven oder auf Grund von Defekten unkooperativ. Verfahren, die Fehlverhalten in stabilen Netzen durch direkte Beobachtung erkennen können, sind in VDTNs nicht anwendbar. Daher sind Methoden, die Fehlverhalten in VDTNs nachweisen können, zwingend erforderlich. In dieser Arbeit werden zwei Klassen von Misbehavior Detection Systems (MDSs) vorgestellt. Beide Systeme basieren auf Encounter Records (ERs): Nach einem Kontakt tauschen zwei Knoten kryptografisch signierte Meta-Informationen zu den erfolgten Datentransfers aus. Diese ERs dienen bei darauffolgenden Kontakten mit anderen Netzwerkteilnehmern als vertrauenswürdiger Nachweis für das Verhalten eines Knotens in der Vergangenheit. Basierend auf der Auswertung gesammelter ERs wird ein Reputationssystem entwickelt, das kooperatives Verhalten belohnt und unkooperatives Verhalten bestraft. Dauerhaft unkooperative Knoten werden aus dem Netzwerk ausgeschlossen. Durch den asynchronen Austausch von Informationen kann jeder Knoten das Verhalten seiner Nachbarn selbstständig und unabhängig evaluieren. Dadurch sind die vorgestellten MDS-Varianten sehr gut für den Einsatz in einem VDTN geeignet. Durch umfangreiche Evaluationen wird gezeigt, dass sich die entwickelten MDS-Verfahren für verschiedene Routingprotokolle und in unterschiedlichen Szenarien anwenden lassen. In allen Fällen ist das MDS in der Lage das System mit geringem Overhead gegen Angreifer zu verteidigen und eine hohe Servicequalität im Netzwerk zu gewährleisten.

Acknowledgments

First of all, I offer my sincerest gratitude to my supervisor, Prof. Lars Wolf, who has supported me throughout my thesis with his patience and knowledge whilst allowing me the room to work in my own way. He is not only an excellent professor, but also the best friend for me, who always gives available support and encouragement to me. Moreover, I would also like to thank my external examiner Prof. Frank Kargl, who shares his invaluable insight on vehicular security with me.

The time at the Institut für Betriebssysteme und Rechnerverbund (IBR) was always wonderful. First, I would like to thank Sebastian Schildt. It was a wonderful time that we could work collaboratively at the IBR group and had many discussions to brainstorm and review our work. Further, I would like to thank Dr. Tobias Pögel, Johannes Morgenroth, Dr. Felix Büsching, Stephan Rottmann and Dominik Schürmann for their contributions to my research.

I am very grateful to Mareike Schmidt, Antje Lemke, Ulrich Timm and Frank Steinberg who help me have a comfortable and pleasant life in Deutschland.

Finally, I would never get this far without the support of my family. I know they always believe in me, support me and wish for my success. Their love and encouragement have been and will always be a great source of inspiration in my life.

Contents

List of Figures xi

List of Tables xv

1 Introduction 1
- 1.1 Vehicular Delay- and Disruption-Tolerant Networks . . . 1
- 1.2 Motivations and Objectives . . . 2
- 1.3 Research Contributions . . . 4
- 1.4 Outline of the Thesis . . . 5

2 Background and Related Work 7
- 2.1 Applications in VDTNs . . . 7
- 2.2 Overview of DTN Routing Protocols . . . 10
  - 2.2.1 Unlimited-copy Routing Protocols . . . 11
    - 2.2.1.1 Epidemic . . . 11
    - 2.2.1.2 MaxProp . . . 12
    - 2.2.1.3 PROPHET . . . 13
  - 2.2.2 Limited-copy Routing Protocols . . . 13
    - 2.2.2.1 Spray and Wait . . . 13
    - 2.2.2.2 RUTS . . . 14
- 2.3 Encryption and Authentication . . . 15
  - 2.3.1 Public Key Infrastructure . . . 15
  - 2.3.2 Identity-based Cryptography . . . 16
- 2.4 Misbehavior Detection System . . . 18
  - 2.4.1 Reputation Systems . . . 18
    - 2.4.1.1 MDS in MANETs . . . 18
    - 2.4.1.2 MDS in VANETs . . . 21
    - 2.4.1.3 MDS in DTNs . . . 23
  - 2.4.2 Credit Systems . . . 27
  - 2.4.3 Tit-for-Tat Systems . . . 31
- 2.5 Conclusions . . . 32

3 Basic Architecture of Misbehavior Detection System 33
- 3.1 Introduction . . . 33

3.2 System Model . . . 34
3.2.1 Vehicular Node Model . . . 34
3.2.1.1 Encounter Record . . . 35
3.2.1.2 Meeting List . . . 37
3.2.1.3 Local Blacklist . . . 38
3.2.1.4 Friend Blacklist . . . 39
3.2.2 Attack Model . . . 39
3.2.2.1 Basic Attacks . . . 41
3.2.2.2 Advanced Attacks . . . 41
3.3 System Architecture . . . 42
3.3.1 Rule Violation Checks . . . 43
3.3.2 Evaluation Module . . . 45
3.3.2.1 Trust Reputation . . . 45
3.3.2.2 TR Update Principle . . . 46
3.3.2.3 Upper Bounding of the *Re* Set . . . 48
3.3.3 Decision Module . . . 51
3.3.3.1 FBL Principle . . . 52
3.4 Performance Evaluations . . . 52
3.4.1 The ONE Simulator . . . 53
3.4.2 Evaluation Metrics . . . 53
3.5 Conclusions . . . 55

4 Adaptive Threshold-based Approach to Detect Attackers in Vehicular DTNs 57
4.1 Introduction . . . 57
4.2 Evaluation Module . . . 57
4.2.1 Update Ratios . . . 58
4.2.2 TR Update . . . 59
4.2.3 Threshold Strategy . . . 61
4.2.3.1 Fixed Threshold . . . 61
4.2.3.2 Adaptive Threshold . . . 62
4.3 Performance Evaluations . . . 63
4.3.1 Simulation Setup . . . 63
4.3.2 Using Fixed Threshold to Cope with Attackers . . . 64
4.3.2.1 Detection Rate . . . 64
4.3.2.2 False Positive Rate . . . 66
4.3.2.3 Detection Speed . . . 67
4.3.3 Using Adaptive Threshold to Cope with Attackers . . . 68
4.3.3.1 Detection Rate . . . 68
4.3.3.2 Detection Rate with Colluding Attackers . . . 70
4.3.3.3 False Positive Rate . . . 73
4.3.3.4 Detection Speed . . . 74

4.3.3.5 Relayed Messages . . . 75
4.3.3.6 Delivery Rate . . . 77
4.4 Conclusions . . . 82

5 Using Cluster Analysis to Detect Attackers in Vehicular DTNs 83
5.1 Introduction . . . 83
5.2 Evaluation Module . . . 84
5.2.1 Update Ratios . . . 84
5.2.1.1 Applicability of the θ and ψ Parameters for Misbehavior Detection . . . 85
5.2.2 Cluster Analysis . . . 91
5.2.2.1 K-means Clustering . . . 91
5.2.3 Improved TR Update Principle . . . 94
5.2.4 TR Update . . . 95
5.3 Performance Evaluations in Homogenous VDTNs . . . 98
5.3.1 Simulation Setup . . . 98
5.3.1.1 Helsinki Scenario . . . 98
5.3.1.2 San Francisco Scenario . . . 99
5.3.1.3 Braunschweig Scenario . . . 99
5.3.2 Detection Rate . . . 100
5.3.3 Detection Rate with Low Amount of Attackers . . . 104
5.3.4 Detection Rate with Colluding Attackers . . . 108
5.3.5 False Positive Rate . . . 111
5.3.6 Misclassification Rate . . . 115
5.3.7 Detection Speed . . . 117
5.3.8 Relayed Messages . . . 122
5.4 Performance Evaluations in Heterogeneous VDTNs . . . 126
5.4.1 Simulation Setup . . . 127
5.4.2 Pedestrians Supporting the VDTN . . . 128
5.4.2.1 Delivery Rate in the VDTN . . . 128
5.4.2.2 Latency in the VDTN . . . 130
5.4.2.3 Overhead Rate in the VDTN . . . 131
5.4.3 Vehicles Supporting the Smartphone-based DTN . . . 132
5.4.3.1 Delivery Rate and Latency in the Smartphone-based DTN . . . 133
5.4.3.2 Overhead in the Smartphone-based DTN . . . 133
5.4.4 MDS Performance . . . 135
5.5 Conclusions . . . 138

6 Conclusions 141
6.1 Review of Contributions . . . 141
6.2 Future Directions . . . 143

Bibliography **145**

List of Figures

1.1 Vehicular DTNs 2

2.1 DakNet 8
2.2 KioskNet 9
2.3 DieselNet 9
2.4 EMMA 10
2.5 Node A Exchanges a Message with Node B Using Epidemic 11
2.6 Node A Exchanges a Message with Node B Using MaxProp 12
2.7 Node A Exchanges a Message with Node B Using PROPHET 13
2.8 Public Key Infrastructure (PKI) 16
2.9 Identity-based Cryptography (IBC) 17
2.10 Watchdog 19
2.11 CONFIDANT 20
2.12 Defense Scheme Architecture 22
2.13 Ferry Patrol Route 23
2.14 Encounter Ticket 25
2.15 Dropping Detection 26
2.16 One-level Ad Dissemination 29
2.17 Layered Coin Architecture 30

3.1 The Process of Encounter Record Exchange 36
3.2 The Process of Signing Encounter Record 37
3.3 Comparison of Different Strategies to Update TR 47
3.4 The Distribution of Connectivity Time 49
3.5 Gamma Distribution 51

4.1 Variable Detection Threshold 63
4.2 Detection Rate – Using Fixed Threshold in the Helsinki Scenario . . . 65
4.3 False Positive Rate – Using Fixed Threshold in the Helsinki Scenario 67
4.4 Detection Speed – Using Fixed Threshold in the Helsinki Scenario . . 68
4.5 Detection Rate – Using Adaptive Threshold in the Helsinki Scenario . 70
4.6 Detection Rate for Blackholes with and without Collusion – Using Adaptive Threshold in the Helsinki Scenario 72
4.7 False Positive Rate – Using Adaptive Threshold in the Helsinki Scenario 73

4.8 Detection Speed – Using Adaptive Threshold in the Helsinki Scenario 75
4.9 Relayed Messages – Using Adaptive Threshold in the Helsinki Scenario 77
4.10 Delivery Rate of the MDS Using Adaptive Threshold in the Helsinki Scenario . . . 79
4.11 Delivery Rate of the MDS with the FBL Using Adaptive Threshold in the Helsinki Scenario . . . 80
4.12 Delivery Rate of Epidemic with Constrained Buffer Size Using Adaptive Threshold in the Helsinki Scenario . . . 81

5.1 (θ, ψ) Values Using Epidemic in the Helsinki Scenario . . . 86
5.2 (θ, ψ) Values Using MaxProp in the Helsinki Scenario . . . 87
5.3 (θ, ψ) Values Using PROPHET in the Helsinki Scenario . . . 88
5.4 (θ, ψ) Values Using Spray and Wait in the Helsinki Scenario . . . 89
5.5 K-means Algorithm . . . 93
5.6 Using Basic Principle of TFT to Update TR . . . 95
5.7 Detection Rate – Using Cluster Analysis in the Helsinki Scenario . . . 102
5.8 Detection Rate – Using Cluster Analysis in the San Francisco Scenario 103
5.9 Detection Rate – Using Cluster Analysis in the Braunschweig Scenario 104
5.10 Detection Rate with a Single Attacker – Using Cluster Analysis in the Helsinki Scenario . . . 105
5.11 Detection Rate with a Single Attacker – Using Cluster Analysis in the San Francisco Scenario . . . 107
5.12 Detection Rate with a Single Attacker – Using Cluster Analysis in the Braunschweig Scenario . . . 108
5.13 Detection Rate for Blackholes with and without Collusion – Using Cluster Analysis . . . 110
5.14 False Positive Rate – Using Cluster Analysis in the San Francisco Scenario . . . 113
5.15 False Positive Rate – Using Cluster Analysis in the Braunschweig Scenario . . . 114
5.16 Misclassification Rate – MDS Using Cluster Analysis . . . 116
5.17 Detection Speed – Using Cluster Analysis in the Helsinki Scenario . . . 119
5.18 Detection Speed – Using Cluster Analysis in the San Francisco Scenario 120
5.19 Detection Speed – Using Cluster Analysis in the Braunschweig Scenario 121
5.20 Relayed Messages – Using Cluster Analysis in the Helsinki Scenario . . . 123
5.21 Relayed Messages – Using Cluster Analysis in the San Francisco Scenario 124
5.22 Relayed Messages – Using Cluster Analysis in the Braunschweig Scenario 125
5.23 The DTN Hardware for a Tram . . . 127
5.24 Delivery Rate in the VDTN . . . 129
5.25 Latency in the VDTN . . . 131
5.26 Overhead Rate in the VDTN . . . 132
5.27 Delivery Rate in the Smartphone-based DTN . . . 134

5.28 Latency in the Smartphone-based DTN 134
5.29 Overhead in the Smartphone-based DTN 135
5.30 Malicious Vehicle Detection Rate with and without Pedestrians . . . 136
5.31 Detection Rate in the Hybrid VDTN 138

List of Tables

3.1 Format of the Meeting List 38
3.2 Format of the Local Blacklist 39
3.3 Format of the Friend Blacklist 39

4.1 Simulation Parameters in the Helsinki Scenario 64

5.1 Simulation Parameters in the San Francisco Scenario 99
5.2 Simulation Parameters in the Braunschweig Scenario 100
5.3 Simulation Parameters in the Hybrid Braunschweig Scenario 128

Chapter 1

Introduction

1.1 Vehicular Delay- and Disruption-Tolerant Networks

In vehicular ad hoc networks (VANETs) [1], many vehicular nodes participate in a dynamic wireless network and have messages for each other. A vehicular node equipped with short-range radios has the ability to communicate with fixed roadside infrastructures, gateways or other nearby vehicular nodes. Today the typical communication application in VANETs is to exchange messages among neighboring vehicles. With the recent advances of technology, it has become feasible to equip almost any device with wireless networking capabilities. In the future this may be more generic including communication from vehicles to arbitrary communication partners.

However, in VANETs persistent connectivity among vehicular nodes cannot be guaranteed everywhere. Due to the high mobility of vehicular nodes, the connectivity in vehicular networks is highly unstable, links may change or break soon after they have been established and the network topology varies significantly depending on time and location. Especially when the network exhibits scarce transmission opportunities and intermittent connectivity, it is difficult to forward messages to the destinations. Delay- and Disruption-Tolerant Networks (DTNs) [2] are suitable for diverse applications when the network suffers from intermittent connections and significant delays. DTNs implement a "store, carry and forward" paradigm to handle conditions where a continuous end-to-end link may not always be available and intermittent connectivity and significant delays are the norm. A packet will be sent over an existing link and buffered at a node until a connection to a suitable next hop is established. By using the store-carry-forward paradigm, the packet moves along a path in a hop-by-hop fashion with possibly long stop on some nodes until it eventually reaches the destination. At this early stage of commercialized VANET communication and until a high penetration of networked vehicles is realized, at least

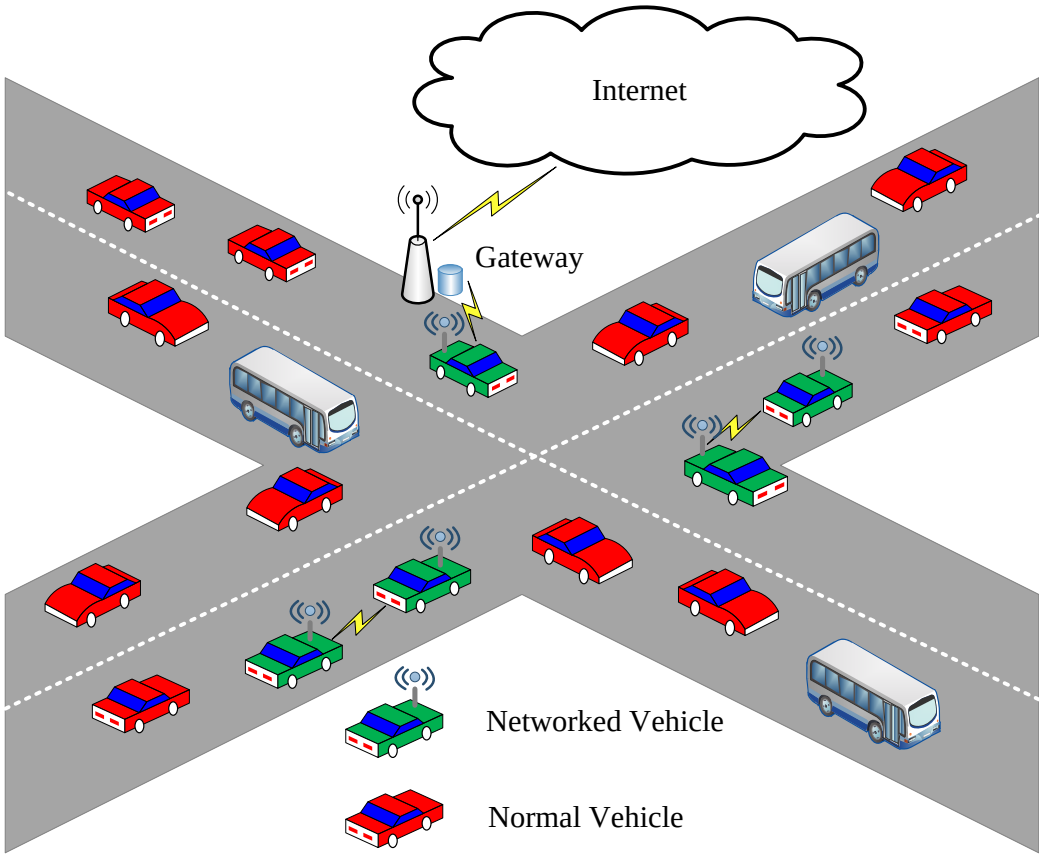


Figure 1.1: **Vehicular DTNs**

for the next years, delay-tolerant methods are a necessity or at least useful extension in such networks, leading to Vehicular Delay- and Disruption-Tolerant Networks (VDTNs) (see Figure 1.1). VDTNs do not only extend the capability of VANETs enabling communication in both dense and sparse networks but they can also support other classes of networks such as Wireless Sensor Networks (WSNs), Pocket Switched Networks (PSN) or the Internet. Several projects have already been implemented for VDTNs (see section 2.1).

1.2 Motivations and Objectives

Many proposed vehicular network systems are based on the hypothesis that vehicular nodes cooperate to forward messages and no abnormal behavior from participating vehicular nodes occurs. However, vehicular nodes are individual entities that can make independent decisions regarding the forwarding or deletion of messages. Some of the vehicular nodes may be malicious, trying their best to destroy or disrupt

the network. Additionally, considering that the communication systems of vehicular nodes are realized by using widely available commodity hard- and software, such as Wi-Fi and standard operating systems, attackers can penetrate these systems and compromise a node in order to alter its behavior and attack the network. Especially if multiple attackers exist and collude to disrupt the network, this can cause severe problems in VDTNs. If we can detect and punish malicious behavior, good nodes can avoid exchanging messages with offenders, which lessens the network load and improves the network performance as reliable paths are chosen for the messages. Besides, some of the vehicular nodes are not altruistic in such a network. Considering energy and memory, selfish nodes may be reluctant to cooperate if it is not directly beneficial to them. Therefore, selfish nodes should be encouraged to forward other nodes' messages by giving them incentives. Another common failure mode is generated by hard- or software errors. Hard- or software errors often manifest themselves in lost messages. Hence, if there are faulty vehicular nodes which cannot send or forward data, the system wants to detect them as fast as possible to prevent the loss of messages.

Defending the VDTNs against the interference of malicious, selfish and faulty nodes, security consideration is clearly an important issue. Some work has focused on authentication and encryption [3, 4, 5, 6, 7, 8, 9, 10]. Authentication and encryption are efficient methods to defend the network against unauthorized outside attackers. However, authorized inside attackers can still launch the attacks to the network and pose severe threats for the network. In other words, authentication and encryption are a necessary first line of defense but they can neither safeguard the system from inside attackers, nor guarantee the willingness of nodes to cooperate. Hence, a flexible Misbehavior Detection System (MDS) is essential for VDTNs.

A typical MDS approach [11, 12, 13, 14] is to observe the behavior of nodes such as their data forwarding. However, because of the opportunistic connections, significant delays and high mobility of nodes in VDTNs, it is difficult to observe data forwarding directly. Some recent work has suggested the usage of encounter tickets to tackle the problem of misbehavior detection in DTNs [15, 16, 17]. When a node encounters another node, these tickets will be exchanged. An encounter ticket contains mutually agreed information regarding a contact between two nodes and can later be used as proof of a node's behavior in the past. However, these systems are designed to only detect blackhole attackers. Also, when attackers with the ability to forge encounter tickets exist, the system in [17] needs to rely on additional nodes in range to make a correct decision. Besides, these approaches are tied to a specific routing protocol, which limits their applicability to specific scenarios.

Detecting attackers is difficult in a network without continuous connectivity where end-to-end connectivity can not be guaranteed. Since reliability and resilience are important factor in VDTNs, in this thesis we will propose a MDS that enables nodes in

VDTNs to independently detect attackers by distributing and combining information from previous encounters and prevent attackers from having a devastating effect on network performance. Our MDS will detect suspicious information and identify the behavior of nodes. Attackers will be banished from the network as fast as possible. If selfish nodes implement non-cooperation, they will get a lower service quality from the network. Besides, the system will detect faulty nodes to avoid useless transmission. Our MDS will focus on detecting authorized inside attackers. However, how to issue the certificate to nodes and encrypt messages between nodes is out of the context of this thesis.

1.3 Research Contributions

We propose a general mechanism to detect unwanted behavior and encourage cooperation of nodes in VDTNs. By collecting and securely exchanging data of previous encounters, a node can assess the trustworthiness of other nodes in order to detect malicious behavior. Our MDS is designed to detect faulty nodes, selfish nodes, blackhole attackers and greyhole attackers with varying drop probabilities (see section 3.2.2). We evaluate our method in 3 different metropolitan scenarios using different DTN routing protocols. Specifically, we make the following contributions:

Depending on the amount of information available from asynchronous exchange of observations through encounter records, the system will adaptively choose a suitable detection threshold to maximize detection rates while minimizing false positive rates. The strategy of adaptive threshold can work in different scenarios where the number of malicious nodes, the attack intensity or the employed routing protocol is varied.

We extend the threshold-based MDS with the idea of cluster analysis, which allows the system to adapt to the current situation and better discriminate between good and malicious behavior. First, the clustering method enables the MDS to defend a VDTN against attackers without the need of an initial learning phase. Secondly, compared to the adaptive threshold mechanism the cluster analysis only uses a few scenario-independent constants as parameters. The wide difference among the evaluated scenarios is bridged by the dynamic cluster analysis, which can make the system perform well in different scenarios without the need to train and fine-tune the system to a specific scenario. Thirdly, the MDS is very generic and it does not rely on any specific routing protocols. Most importantly, using the cluster analysis the MDS does not only defend homogenous vehicular networks but also hybrid networks, where vehicular nodes are mixed with pedestrian nodes.

Our system introduces an incentive mechanism to encourage the cooperation of nodes. Depending on the behavior of vehicular nodes, different trust reputation

will be given to nodes. Cooperative nodes, which prefer forwarding messages for other nodes, will obtain a good trust reputation and be contacted first compared to uncooperative nodes. Uncooperative nodes, with a very low trust reputation, will get a lower service quality from the network compared to cooperating nodes or even be excluded from the network.

Nodes mutually keeping a good reputation for each other will become friend nodes. We utilize the friend relation to share more information regarding the assessment of other nodes. The results in our thesis show that the friend mechanism can speed up the detection process, making the MDS achieve a high detection rate and a low false positive rate.

1.4 Outline of the Thesis

The remainder of the thesis is structured as follows.

Chapter 2 provides detailed information about the background of this thesis. This chapter includes an overview of recent VDTN projects using delay-tolerant methods as well as widely used routing protocols which are designed for DTNs. Moreover, we survey the security strategies in DTNs, such as authentication and encryption to defend against unauthorized outside attackers and MDSs to cope with authorized inside attackers.

Chapter 3 explains our architecture and detection scheme. We introduce the used vehicular node model and the attack model. Afterwards, we propose the three basic modules of our MDS architecture. The basic principles of the MDS are discussed and a reputation system is proposed in this chapter. In addition, we present our simulator and evaluation metrics to evaluate our MDS.

Chapter 4 proposes a MDS using the fixed threshold and the adaptive threshold to defend a VDTN against attackers. We describe the strategy of the fixed threshold and the adaptive threshold in detail. The simulation-based evaluation shows that our MDS can efficiently detect attackers using different DTN routing protocols.

Chapter 5 proposes a general mechanism using cluster analysis to assess the trustworthiness of other nodes in order to detect malicious behavior. We describe how cluster analysis works. Moreover, we propose an improved trust reputation update principle for the reputation system. The extensive evaluation demonstrates that the proposed MDS is feasible and able to efficiently protect the homogenous and hybrid system with low overhead.

Chapter 6 draws our conclusions, summarizes the main achievements of the thesis and gives some insight into future work.

Chapter 2

Background and Related Work

2.1 Applications in VDTNs

DTNs are a suitable technology for many applications when the network suffers from intermittent connections and significant delays. Delay-tolerant methods were initially used in Interplanetary Internet [18], where space communication may suffer from very large latency and intermittent scheduled connectivity. It has been proposed that DTNs can be used widely, e.g. in wildlife tracking sensor networks [19, 20, 21], military networks [22, 23, 24], inter-planetary networks [25, 26, 27, 28], nomadic communities networks [29, 30] and public transport networks [31, 32, 33, 34] etc..

Nowadays vehicles are becoming increasingly intelligent and the majority will soon be equipped with short-range radios and capable of communicating with roadside infrastructures, gateways or other vehicles nearby. This will allow vehicles to be an enabler for a wide range of applications including real-time traffic monitoring, information sharing, environment monitoring and interactive communications between vehicles. Using the idea of DTNs, some special applications have already been implemented for vehicular networks.

DakNet [31, 35] proposes a store-and-forward vehicular network for sparse connectivity in a rural environment lacking digital communication infrastructure. DakNet consists of transport vehicles equipped with WiFi radio transceivers, stationary kiosk-terminals and an Internet hub. An available transport vehicle, i.e. a bus or a motorcycle, will move along a predefined path and regularly traverse a series of villages. The transport vehicles will transmit data over short point-to-point links with kiosk-terminals when they come into kiosk-terminals' communication range. The kiosk-terminals are responsible for gathering data and then upload or download the data with transport vehicles. The Internet hub provides an Internet access to transport vehicles. By implementing the asynchronous communication infrastructure, DakNet

can provide the service such as e-mail, transfer of educational materials, public health announcements or news etc. to rural areas (see Figure 2.1).

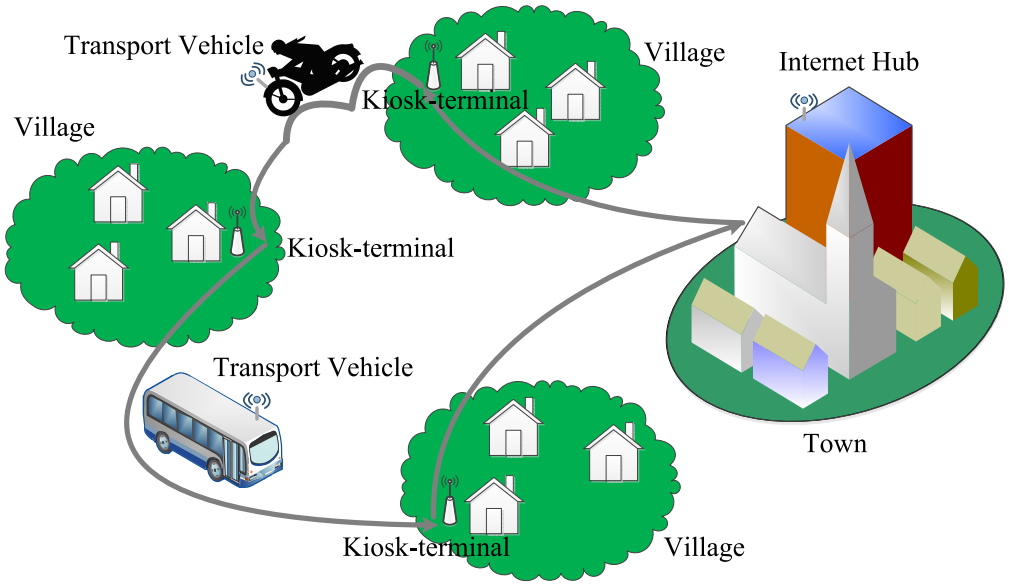


Figure 2.1: **DakNet [35]**

A similar project is KioskNet [32, 36] (see Figure 2.2). Utilizing buses, motorcycles and cars to carry data between rural village kiosks and Internet gateways, KioskNet provides very low-cost Internet services to developing regions. Compared to DakNet, besides using WiFi as the primary mode of communication, KioskNet also supports multiple network interfaces, i.e. a cellular or dial-up connection, at each kiosk. And to minimize redundant data which transfers during an opportunistic connection, a special DTN protocol is used in KioskNet to improve the effectiveness of transferring data in the system. Moreover, KioskNet leverages VDTNs for disconnection tolerance and considers security problems such as confidentiality and integrity.

DieselNet [33, 37] evaluates the performance of bus-to-bus and bus-to-infrastructure communication in DTNs with the help of buses equipped with off-the-shelf communication hardware (see Figure 2.3). Campus buses supporting WiFi, 3G and GPRS radio technologies, periodically traverse between a disruption tolerant network and a dense mobile network and communicate with each other. To create additional contact opportunities among buses, throwboxes are put in flexible places in the system to act as stationary relays. Dedicated access points are used to connect to the Internet. With fixed mobility patterns, buses gather information and exploit open access points around the campus to upload data to a central data server. Building such a versatile testbed, DieselNet can support research results for evaluating the communication performance in vehicular networks.

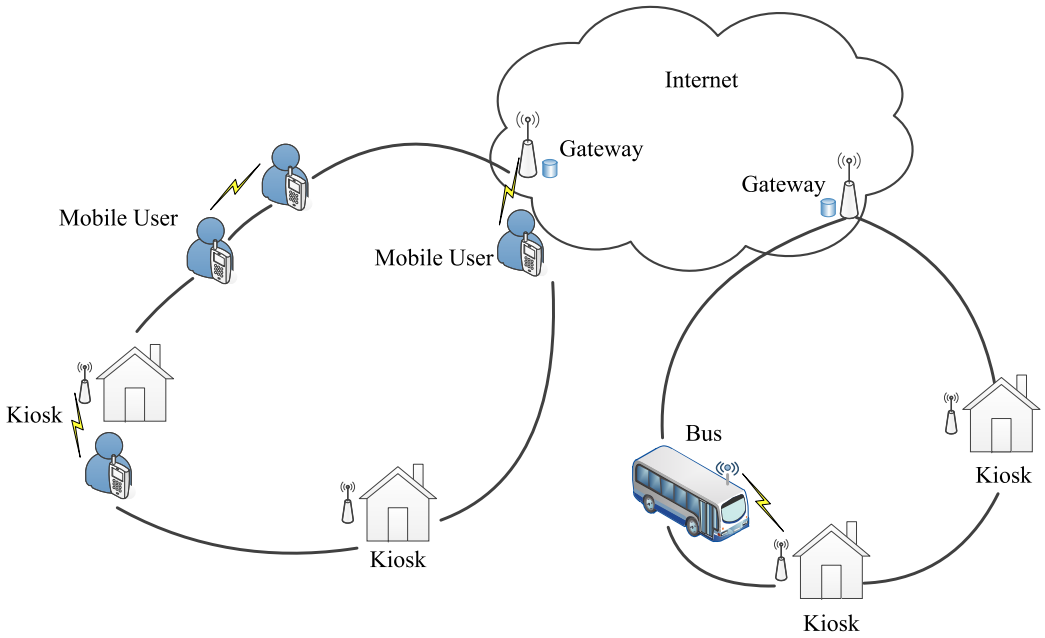


Figure 2.2: **KioskNet [32]**

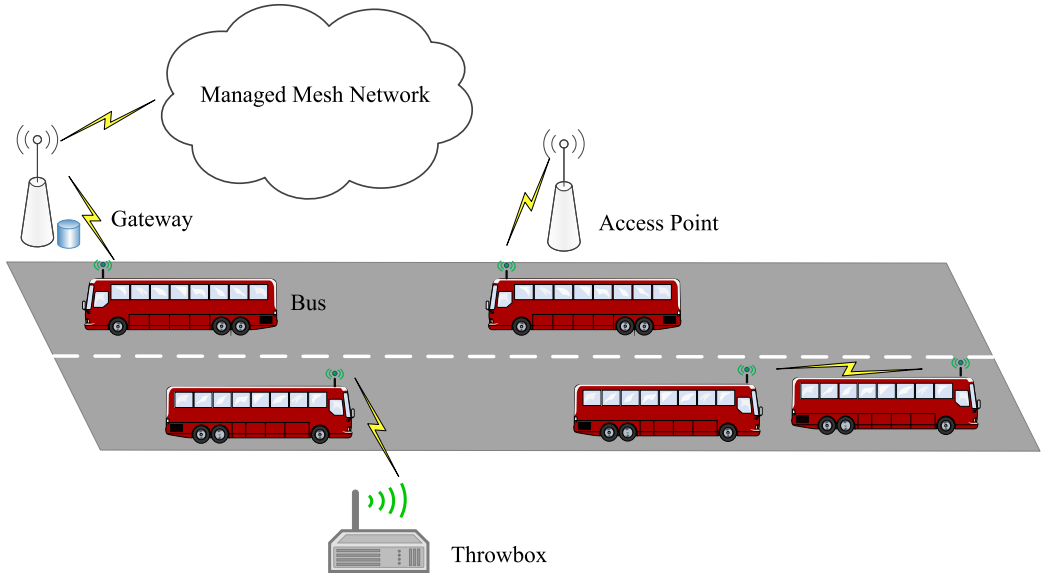


Figure 2.3: **DieselNet [37]**

In the project Environmental Monitoring in Metropolitan Areas (EMMA) [34, 38], the public transport system composed of buses and trams is used to gather air pollution measurements (see Figure 2.4). Real-hardware was developed and deployed to transfer air pollution data by using the IBR-DTN Bundle Protocol implementation [39]. Buses and trams move within the whole metropolitan area, continuously collect the air pollution data and send the data to a gateway. The gateway is a sta-

tionary node which is responsible for gathering the air pollution data and sends it to the data center for further analysis. By using delay-tolerant methods, EMMA builds a decentralized architecture for environmental monitoring. Additionally, the system is extended to introduce other applications such as transporting timetables or tourist information to display at bus and tram stops. Detailed information about EMMA is introduced in section 5.3.1.3. Moreover, we will use EMMA as an exemplary VTDN scenario and assure the security of EMMA.

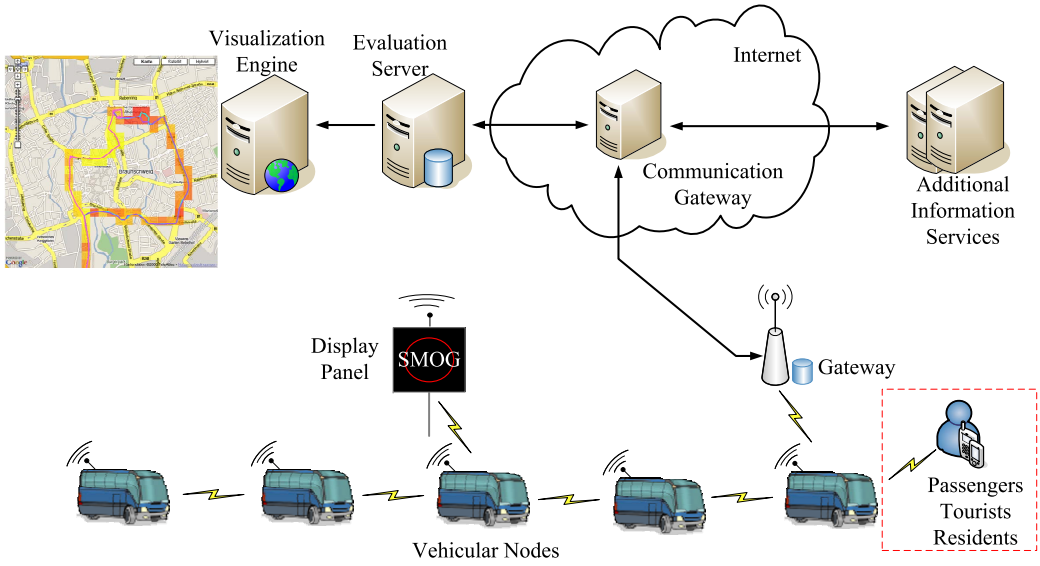


Figure 2.4: EMMA [34]

These examples show that VDTNs are a very promising communication system. VDTNs will provide versatile applications until a high penetration of networked vehicles is achieved [40, 41]. Despite having a lot of potential, vehicular networks are also one of the most challenging networks. Considering the dynamic network topology based on the mobility of the vehicles and the environmental impact on the radio propagation, the vehicles should not waste any contact opportunities but try to forward their data whenever there is an opportunity. Hence, in this thesis we will specially focus on the security issue of VDTNs and make the vehicles more effectively utilize the precious opportunities.

2.2 Overview of DTN Routing Protocols

Routing protocols are one of the key components in VDTNs [42, 43]. Facing the dynamic network topology, how to build a good path from the source to the destination is the major responsibility for routing protocols. Especially in VDTNs, persistent connectivity among vehicular nodes cannot be guaranteed everywhere. There are no

end-to-end contemporaneous paths in the network and meanwhile vehicular nodes only hold limited amount of knowledge about the network. Based on different applications of VDTNs, different routing protocols are proposed for VDTNs. In the following section, we survey some selected routing protocols which are widely applied in VDTNs. For each protocol, we describe how the protocol works, and then give a short review for the protocol.

2.2.1 Unlimited-copy Routing Protocols

2.2.1.1 Epidemic

Epidemic [44] is an unlimited-copy routing protocol. This protocol is a simple and effective routing method which extends the concept of flooding. The basic idea is that nodes will transfer messages to any node they encounter.

Each node keeps an index of messages in its memory, called the summary vector. Whenever two nodes meet, they will first exchange their summary vectors to determine whether some messages are unknown by one of them. Using the summary vector, the system can make sure that new messages are exchanged between nodes and moreover the system can prevent unnecessary transmissions of messages which both nodes have already buffered in their memory (see Figure 2.5). Using Epidemic, messages will eventually spread through the whole network as nodes encounter and "infect" each other. Due to its good efficiency, Epidemic becomes one of the most popular routing protocols used in VDTNs.

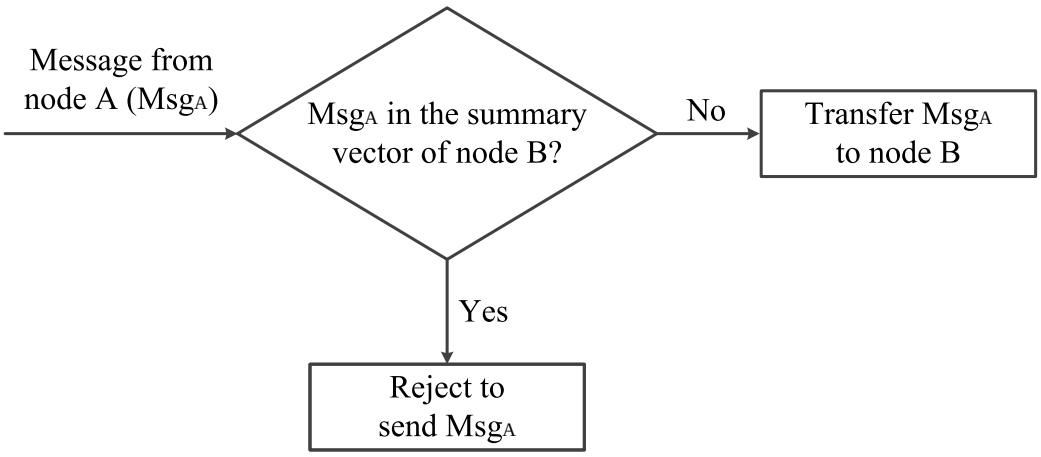


Figure 2.5: **Node A Exchanges a Message with Node B Using Epidemic**

Epidemic is a robust solution to adapt to different scenarios in VDTNs and can achieve low delivery latency and a high delivery probability. However, it comes with a high network load especially with a large number of replicated messages.

Meanwhile, when the nodes' buffer space and the network capacity are limited, the network will face the problem of huge message dropping and retransmissions. Some work [45, 46, 47, 48, 49] has been done to improve the performance of Epidemic, balancing the network load with the delivery rate.

2.2.1.2 MaxProp

MaxProp [33] is a protocol designed for vehicular scenarios. Compared to Epidemic, MaxProp adopts the delivery optimization strategy in DTNs. In this protocol (see Figure 2.6), the authors use two metrics, the message hop count and the delivery likelihood to destinations, to calculate the priorities of messages transmitted to other nodes and deleted due to nodes' full buffer. When a contact emerges, the messages assigned a higher priority are transferred first with higher priority and the messages with a lower priority would be the first to be discarded if needed. Moreover, each node maintains an estimation of the probability of encountering other nodes. The estimation data is used to calculate the path cost of messages. The message's potential paths to the destination will be chosen if the message's path cost is below a threshold. Besides, in MaxProp, after a message has been received by destination node, the destination will send acknowledgments about this message to clear the remaining copies of the message in the network.

Although MaxProp is an unlimited-copy routing protocol, compared to Epidemic, MaxProp does not only improve the delivery rate of the network but also reduces the number of accumulated messages in buffer spaces with the help of acknowledgments.

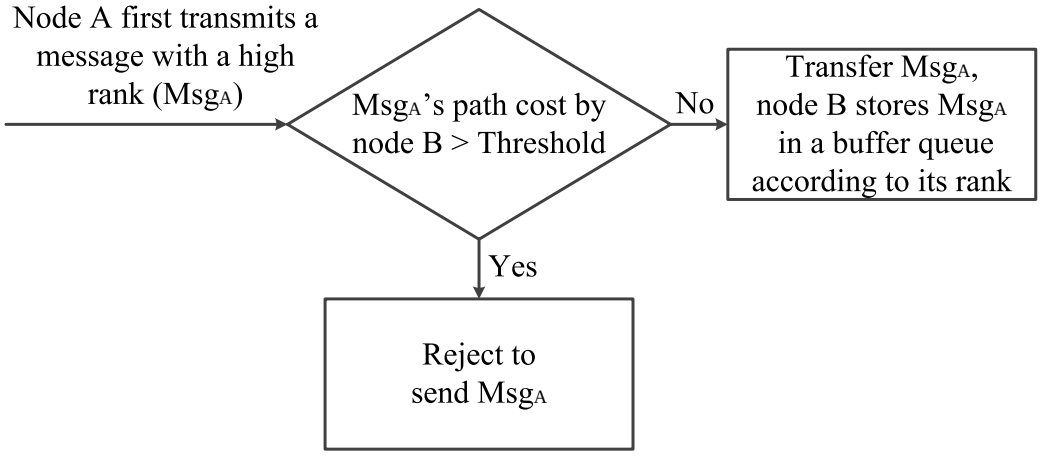


Figure 2.6: **Node A Exchanges a Message with Node B Using MaxProp**

2.2.1.3 PROPHET

Lindgren et al. present a routing protocol called PROPHET [50]. Based on historic encounters, PROPHET establishes a probabilistic metric called the message delivery predictability. When two nodes encounter, the node will send its messages to the other node only when the other node's message delivery predictability is higher than its own message delivery predictability to the destination (see Figure 2.7). In other words, messages are replicated only to the nodes with a better message delivery predictability. PROPHET is an unlimited-copy routing protocol, however, the resource utilization is reduced by sending only a few message copies to nodes with a higher delivery predictability. Hence, the performance of the network is enhanced.

Additionally, PROPHET is designed to cope with non-random mobility models. When nodes with a repeating behavioral pattern exist in the network, PROPHET has the ability to predict good forwarding nodes and keeps the performance of the network well. However, PROPHET has difficulties coping with randomly moving nodes. A similar approach is proposed in the MORA routing protocol [51] and some improved work about PROPHET has been done in [52, 53, 54].

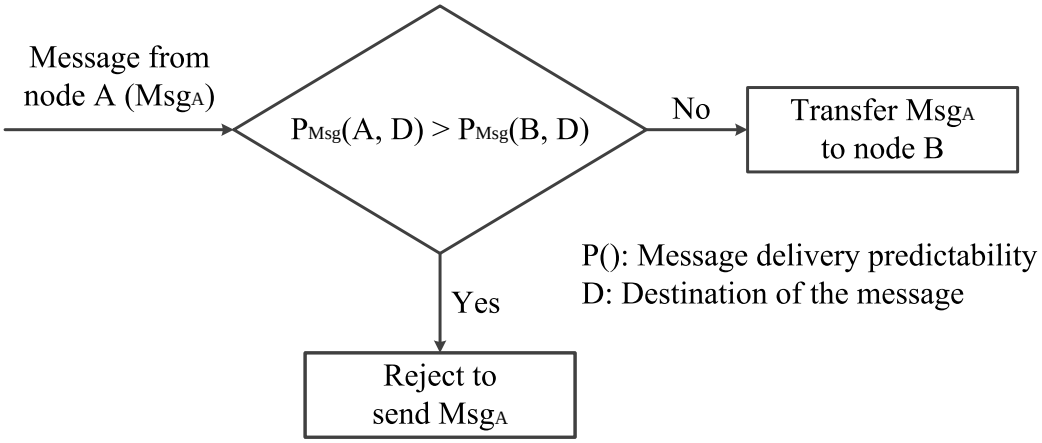


Figure 2.7: **Node A Exchanges a Message with Node B Using PROPHET**

2.2.2 Limited-copy Routing Protocols

2.2.2.1 Spray and Wait

Spray and Wait [55] is a limited-copy routing protocol with two phases: (1) Spray phase, where a source node generates a message and spreads a fixed number of copies of this message to its encountered nodes. (2) Wait phase, where each node carrying a copy of the message performs a direct delivery to the destination. There are two

ways to implement the spray phase: the normal mode and the binary mode. In the normal mode, in each encounter the source node gives one copy of the message to its encountered nodes that do not have the message. In the binary mode, in each encounter the source node forwards half of its message copies to the new encountered node, and then the encountered node keeps spreading these message copies using the same way until the node keeps only one copy in its buffer and changes to the wait phase. Considering that the binary mode obtains less expected delay compared to the normal mode, we apply the binary mode when we use Spray and Wait in our VDTNs.

Compared to the unlimited-copy routing protocols, implementing a mechanism that first distributes a fixed and limited number of copies and then waits for one of the relays encountering the destination, Spray and Wait overcomes the shortcoming for the large number of message transmissions and achieves relatively good delays. However, Spray and Wait requires that the nodes should uniformly distribute and frequently move around the network. Otherwise, it will increase the chance for a message to get "stuck" in the network. Spray and Focus [56] is similar to Spray and Wait. However, in the focus phase, rather than waiting for direct contact with the destination, the copy can still be relayed to a node with higher probability to encounter to the destination.

2.2.2.2 RUTS

RUTS [38] is a DTN routing algorithm designed for urban public transport systems in the project EMMA as introduced in section 2.1. The algorithm exploits the characteristics of urban public transport systems, e.g. timetables and network maps to achieve better routing. Considering the estimated transport time from the sender to the receiver, the number of required hops and the characteristics of each bus line, a limited number of routing paths is chosen when a sender wants to transmit a message. Compared to other routing schemes, RUTS can achieve low latency and high delivery rates and avoid high communication overhead. However, RUTS requires the information of timetables and maps in the network, which limits its scope of application.

Considering the importance of routing protocols, researchers have proposed many different routing protocols for different applications of DTNs e.g. [57, 58, 59, 60]. This thesis will apply Epidemic, MaxProp, PROPHET and Spray and Wait as exemplary routing protocols to support the message transmission in VDTNs.

2.3 Encryption and Authentication

Many proposed vehicular network systems are based on the hypothesis that vehicular nodes cooperate to forward data. However, if there are malicious nodes in the system, it incurs a high risk that these malicious nodes inevitably destroy or disrupt the network. Therefore, it is critical to develop a suite of security mechanisms to protect vehicular networks. Since encryption and authentication [61] can provide data-integrity protection and insure a secure communication session between nodes, encryption and authentication are first brought into consideration, which are efficient methods to defend the system against outside attackers.

2.3.1 Public Key Infrastructure

The Public Key Infrastructure (PKI) [62] has been widely accepted as the best solution to provide a security infrastructure. The traditional PKI utilizes a Certification Authority (CA) to issue cryptographic key certificates which provide secure transactions between senders and recipients through an insecure channel (see Figure 2.8). Each of the legitimate nodes first locally generates its public key and private key ($\langle PK, SK \rangle$) and then requests a corresponding certificate ($Cert$) from the CA. By distributing the certificate containing the public key among the legitimate nodes, an unforgeable and trusted link will be built between senders and recipients. A sender (Alice in Figure 2.8) can establish a secure channel by signing data with its private key ($\langle sign_{SK_{Alice}}(message) \rangle$) and encrypting it with the recipient's (Bob) public key ($\langle encrypt_{PK_{Bob}}(sign_{SK_{Alice}}(message)) \rangle$). When a sender in a network signs a message with its own private key, recipients can verify that the message is generated by the sender using the sender's public key ($\langle verify_{PK_{Alice}}(sign_{SK_{Alice}}(message)) \rangle$). Eavesdropping by adversaries is prevented by encrypting a message with a recipient's public key ($\langle decrypt_{SK_{Bob}}(encrypt_{PK_{Bob}}(sign_{SK_{Alice}}(message))) \rangle$), allowing decryption with the recipient's private key solely.

Many projects apply the PKI to solve the security problems for vehicular communication, such as SEVECOM (SEcure VEhicular COMmunications) [6, 63, 64], NoW (Network on Wheels) [65, 66] and GST (Global System for Telematics)[1] as well as the Car2Car Communication Consortium (C2C-CC)[2]. Considering security requirements such as integrity, confidentiality and non-repudiation etc., the systems use the CA to manage identities and credentials of all nodes. The CA is also responsible for cryptographic key management, privacy protection and secure communication.

[1] `http://www.ertico.com/gst-website/`
[2] `http://www.car-to-car.org/`

The advantage of the PKI is that, even though the CA is invaded by attackers, the private keys of the legitimate nodes are still safe in the system. The PKI is a suitable approach where there exists good connectivity among nodes. However, the use of a PKI approach is difficult in VDTNs which have a high degree of nodes' mobility and frequent network partitioning. Using the PKI, a sender needs to obtain the recipient's public key by an end-to-end round trip. In VDTNs, there is no assurance with regard to the existence of a complete path between two nodes wishing to communicate. It might happen that the sender and the recipient never connect to the same network at the same time. Hence, there exists a huge delay when a sender wants to get the public key of the recipient. Secondly, in the phase of initialization, for some nodes, it could be difficult to connect with the CA to get the certificate, the same problem also happens when new nodes join into the network. And thirdly, certificate revocation is still a challenge because the update in VDTNs will be excessively delayed.

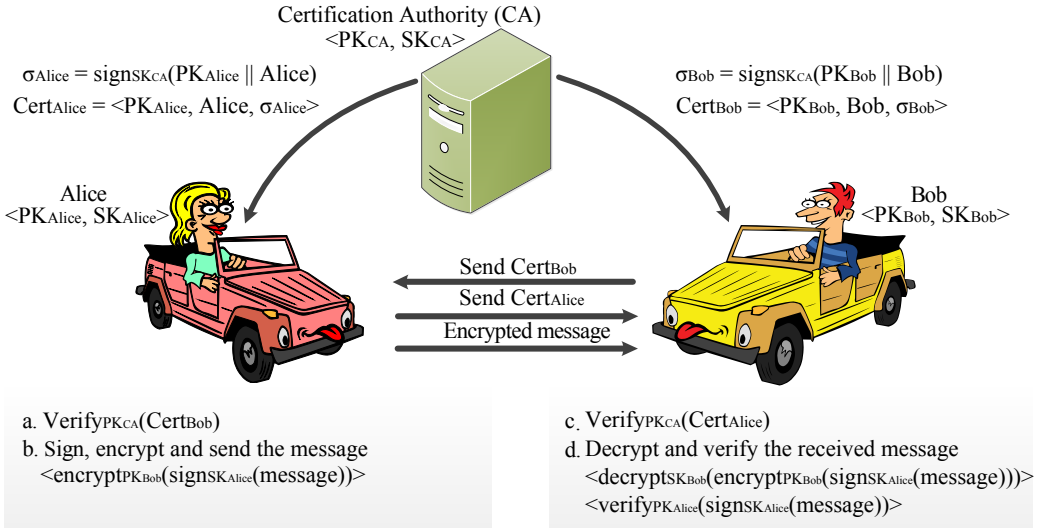


Figure 2.8: Public Key Infrastructure (PKI)

2.3.2 Identity-based Cryptography

Since the PKI does not address the fundamental issue of public key distribution in VDTNs, the idea of an identity-based cryptography (IBC) is proposed in papers [7, 8, 9, 10] as a way of enabling message encryption and signature verification in DTNs. The core idea of the IBC is to make an entity's public key directly derivable from its publicly known identity information.

In the IBC, a node can compute the corresponding public key of the other node by only knowing the other node's identifier (see Figure 2.9). Without relying on any on-line servers or third parties to authorize the certificate of public keys, a node can verify

signatures or encrypt messages for any entity, knowing only its identity information. Using this way, the system reduces the number and frequency of interactions among nodes to obtain others' public keys. In addition, a node needs to request its private key from a trusted Private Key Generator (PKG). A node only needs to contact the PKG once to obtain its own valid private key. Hence, the communication between nodes and the trusted PKG to request valid private keys is asynchronous from the communication among nodes to exchange data. Compared to the PKI, the IBC eliminates the need for public-key certificate transmission and storage. Because of these features, the IBC is particularly suitable for disconnected environments such as VDTNs. However, the big problem for the IBC is that once the attackers invade the PKG, the private keys of the nodes in the whole system are compromised and must be revoked.

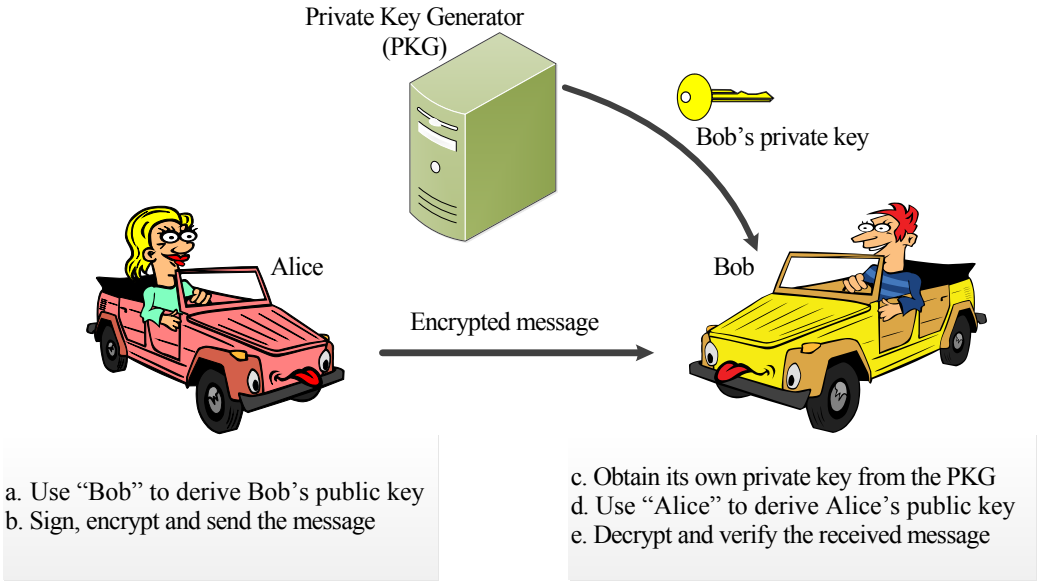


Figure 2.9: **Identity-based Cryptography (IBC)**

The PKI and the IBC both mainly concern the security of unauthorized access and illegitimate utilization of network resources. Compared to the PKI, the adaption of the IBC is more efficient to realize the authentication and provide the security protection in VDTNs.

Considering the advantage of the IBC in terms of the public key distribution, we could use the IBC as the basic cryptographic tool (see section 3.2.1.1) to protect our system and to assure the integrity and confidentiality of the network.

2.4 Misbehavior Detection System

Even though authentication and encryption are efficient methods to defend the system against outside attackers, it cannot safeguard the system from inside attackers, nor guarantee the willingness of nodes to cooperate. Since VDTNs need to be relatively open to be useful, authorized inside attackers are likely threats to the network. Therefore, there is an increasing interest in designing a flexible MDS for VDTNs. The MDS proposes a general mechanism to detect spurious information, identify suspect nodes, mitigate routing misbehavior and correct errors that have been maliciously introduced into data. The mechanism is achieved by analyzing data which is collected by nodes or shared with surrounding neighbors. Ultimately, the MDS can identify a culprit after an attack and prevent the network from being impaired again. Some work has been done in the area of the MDS to detect or mitigate the effects of malicious nodes.

Most existing work falls into one of the following three categories. The first category is based on reputation, where nodes attempt to identify misbehaving nodes and isolate them from the network according to nodes' reputation. The fear of detection and punishment motivates nodes to cooperate. In the second category of solutions, nodes spend credits to obtain forwarding service from any node in the system. Most existing credit-based mechanisms require the support from either a secure hardware or trusted centralized banks. In the third category, nodes reciprocate each other in a tit-for-tat fashion. A node will fully cooperate with a neighbor if no misbehavior is detected, however, it will autonomously lower the service to a neighbor if it detects the misbehavior of the neighbor.

2.4.1 Reputation Systems

Some researchers use the reputation idea to design the MDS. Reputation systems rely on the individual nodes to monitor neighbor nodes' traffic and keep track of each other's reputation. By integrating information, uncooperative nodes can eventually be detected and excluded from the network.

2.4.1.1 MDS in MANETs

In mobile ad hoc networks (MANETs), a typical MDS approach is to observe the behavior of nodes such as their data forwarding. Most of existing MDSs assume that an end-to-end link between the source and the destination exists before data forwarding, therefore, nodes can listen to the next hop's transmissions to detect whether the next hop properly forwards the traffic.

Paper [11] proposes a MDS called Watchdog. In the Watchdog MDS, the watchdog component is responsible for identifying misbehaving nodes, while the pathrater component helps avoid routing packets through these misbehaving nodes.

The nodes use the watchdog component to verify whether the next node in the path forwards the packet (see Figure 2.10). Suppose the source node S sends its packet through intermediate nodes A, B and C to the destination node D. After node A receives the packet and transmits it to node B, node A will keep a copy of this packet in its buffer and promiscuously listen to the transmission of B to make sure that B forwards the packet to node C. If the packet overheard from B matches the packet stored in node A's buffer, node A determines that the behavior of node B is normal, therefore, the watchdog of node A will forget this packet. However, if no matching packet exists after a certain time, the watchdog will increase the failure counter for node B. If the counter exceeds a threshold, node A determines that node B is misbehaving and reports this to the source node S.

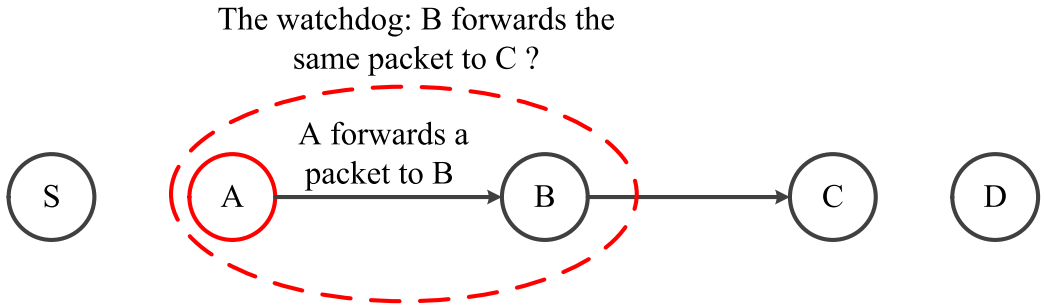


Figure 2.10: **Watchdog [11]**

The pathrater component of nodes uses the information from the watchdog to choose the reliable path to deliver packets. The pathrater component increases the rating of nodes which have sent a packet within the previous interval and decreases a node's rating when the watchdog detects its uncooperative behavior. By integrating the nodes' rating, the pathrater component can finally choose a path with the highest reliability.

Relying on individual nodes to monitor neighbors' traffic, nodes detect the misbehaving nodes and finally avoid the misbehaving nodes in their routes. The simulation of the Watchdog MDS shows that their approach is quite effective for choosing paths to avoid malicious nodes in MANETs. However, because these misbehaving nodes are not punished by the Watchdog MDS, they can save their own resources and meanwhile benefit from the network.

[12, 13] present a reputation-based MDS called CONFIDANT, relying on nodes to monitor neighbors' traffic and keep track of each other's reputation. CONFIDANT eventually detects and excludes malicious nodes from the network.

The monitor, the reputation system, the path manager and the trust manager compose the CONFIDANT MDS (see Figure 2.11). In the monitor, a node uses a "neighborhood watch" to detect malicious behavior within its radio range and ensures that the next surrounding node forwards a packet correctly. If a node detects abnormal behavior of surrounding nodes, the node triggers the reputation system to update the corresponding rating. Once the rating of a surrounding node exceeds the threshold, the path manager will banish the surrounding node from the network and meanwhile makes the trust manager send alarm messages to other neighbors.

When a node receives an alarm message from other nodes in the network, the trust manager will first evaluate the trustworthiness of the alarm message. Only when there is enough evidence to accuse the reported node, the information will be delivered to the reputation system, which performs the same function as described before. And in their later research [13], the authors improve CONFIDANT to balance false accusation and false praise within the system. Compared to the Watchdog MDS, CONFIDANT punishes misbehaving nodes.

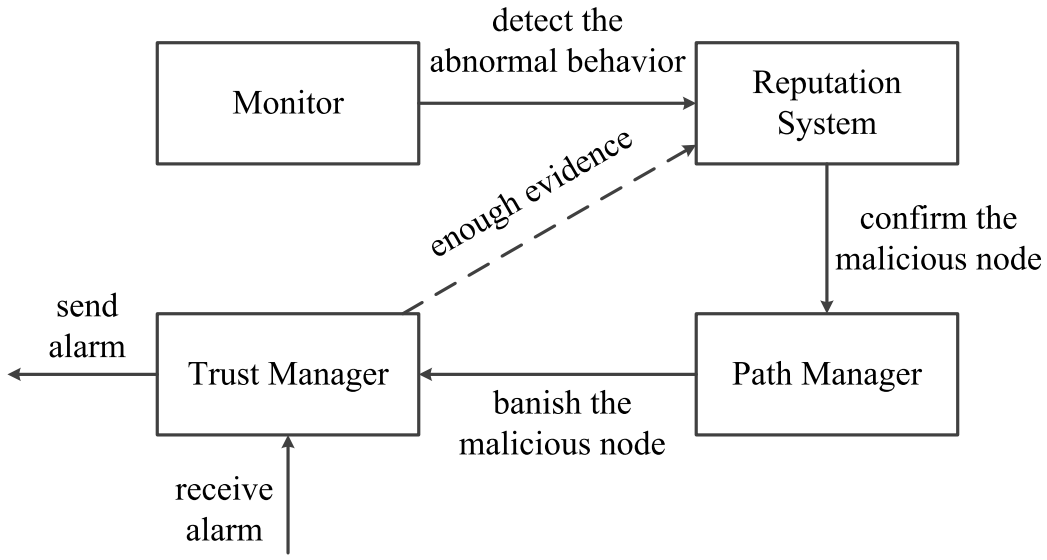


Figure 2.11: **CONFIDANT [12]**

Paper [14] proposes a collaborative reputation-based MDS called CORE to detect selfish nodes. In the CORE MDS, it is requested that the nodes which want to utilize network resources should contribute to routing and forwarding for the system. Hence, by monitoring the behavior of its neighbors, each node holds a reputation value for its neighbors. Once a neighbor's reputation falls below a predefined threshold, the service to the neighbor will be suspended.

The watchdog component of CORE works the same way (as described in paper [11]) to monitor neighbors. The CORE MDS builds its reputation system considering the subjective reputation, the indirect reputation and the functional reputation.

Relying on the direct observations by nodes themselves, the subjective reputation gives a negative rating or a positive rating to the neighbors according to the extent of neighbors' cooperation. The indirect reputation is responsible for sharing the positive reports from nodes. According to the requested function (packet forwarding or route discovery), different functional reputation will be used to combine the subjective and indirect reputation. Compared to the CONFIDANT MDS, the CORE MDS spreads only positive reports, hence prevents the false accusation from disseminating in the whole network.

All these reputation systems assume there exists a full end-to-end path between the source and the destination and the nodes can listen to the next hop's transmissions to detect whether the next hop properly forwards the traffic. However, this assumption does not hold in VDTNs due to its intrinsic opportunistic forwarding nature. In dynamic VDTNs, because of the high mobility of vehicular nodes, nodes are often disconnected and thus it is infeasible to continuously monitor neighbors for detection.

2.4.1.2 MDS in VANETs

Different approaches are proposed to provide the security for VANETs. Except the traditional way using the encryption and authentication idea, some work also describes a defense scheme to detect malicious misbehavior in VANETs.

In VANETs, Golle et al. [67] propose a framework to detect and correct false location claims. In their approach, an individual node maintains a normal model to check the validity of messages from other nodes and searches for possible explanations for each message. When the explanations are consistent with the node's normal model, the nodes which once sent these messages will get scores and have a higher possibility to be accepted by other nodes. When inconsistencies emerge, an adversarial model is used to search for explanations of errors, finally the system tries to correct the consequences of the attack. The approach depends on the cooperation among individual nodes and the normal model of nodes. However, how to build a suitable normal model is still the difficult part for the system.

To address data anomalies, such as modification or injection of messages by legitimate nodes, paper [68] proposes a defense scheme, consisting of a MDS and a LEAVE (Local Eviction of Attackers by Voting Evaluators) and infrastructure-based revocation protocols, to exclude misbehaving vehicular nodes from the network (see Figure 2.12). Vehicular nodes apply the MDS to detect their neighbors. First they use the entropy, a typical measure of information, to detect whether there exists anomalous behavior of nodes. If the entropy exceeds the detection threshold, clustering techniques will be used to distinguish between normal and abnormal behavior and detect

attackers. The MDS enables each individual node to locally detect misbehaving neighbors.

Once a vehicular node detects a misbehaving vehicular node, besides rejecting communication with this misbehaving vehicular node, a warning is triggered and sent out to other vehicular nodes. To prevent wrong accusations, the LEAVE protocol is designed to aggregate warnings and accusations. Only when the number of warnings against a vehicular node exceeds the predefined threshold of the LEAVE, disregard messages are triggered and sent out to the CA. The CA provides the solution to revoke the certificates of misbehaving vehicular nodes. The system assumes that each vehicular node is equipped with a Trusted Component (TC). The TC stores all the necessary cryptographic keys and is capable of cryptographic processing. Hence, the CA can instruct the TC to erase all cryptographic material of the misbehaving nodes, depriving their right to keep working in the network.

The system is built on the assumption that the majority of vehicular nodes is honest. However, this cannot be promised in vehicular networks. If there is a large number of attackers disseminating the warning information, it will give the system a high overhead and affect the detection results of the system.

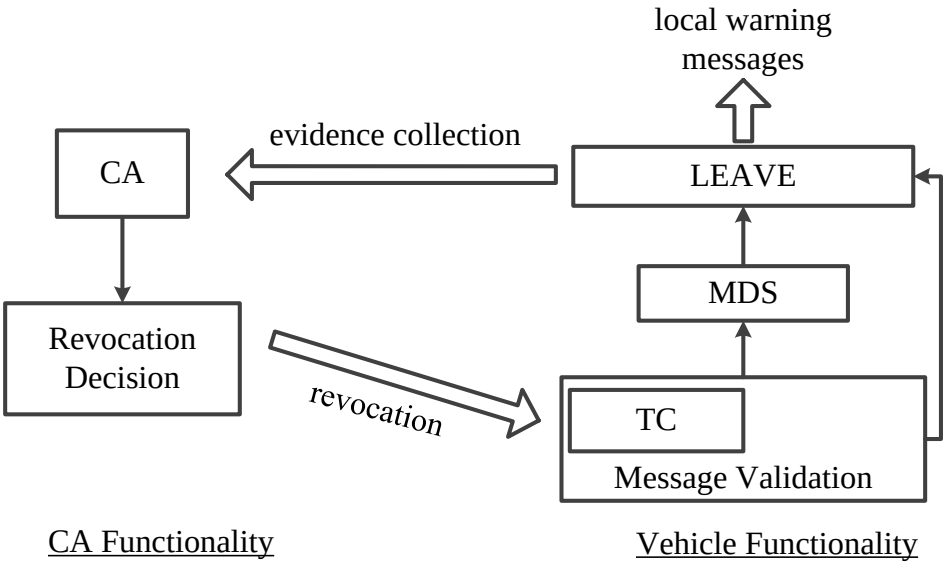


Figure 2.12: **Defense Scheme Architecture [68]**

Some other work about reputation systems in VANETs is proposed in [69, 70, 71]. In paper [69], the authors introduce a distributed Vehicle Behavior Analysis and Evaluation Scheme (VEBAS). VEBAS incorporates several basic checks to detect falsified data.

A vehicular node accumulates a sequence of beacons which are periodically broadcasted by a broadcasting node. These beacons contain the broadcasting node's identifer, its current position and its velocity etc., representing the broadcasting node's past, present and even future movements and its communication activities. Based on these beacons, VEBAS applies multiple behavior analysis modules which can divide into positive-rating modules and negative-rating modules, to form a reputation system. The reputation indicates a broadcasting node's trustworthiness. According to the reputation, a vehicular node can be defined as trustworthy, neutral or untrustworthy. Meanwhile, VEBAS honors evident honest behavior of broadcasting vehicular nodes and disseminates the positive news to all vehicular nodes in the surrounding. Finally, the intentional misbehavior, such as faked position claims, can be detected by VEBAS.

However, VEBAS is still at a conceptual level and lacks technical details and performance evaluation. Moreover, VEBAS requires vehicular nodes to continuously make observations and have a quick response to these beacons, which may increase the overload of vehicular nodes.

2.4.1.3 MDS in DTNs

Recently, a ferry based detection method [72, 73, 74] is proposed to tackle the problem of misbehavior detection in DTNs. The ferry travels along fixed routes and collects the information from the nodes on its path to identify potentially malicious nodes in the network (see Figure 2.13). Once the ferry declares one node as malicious, it will broadcast the news as it travels along its path. Any node hearing such a news will not choose the malicious nodes as the next hop node anymore.

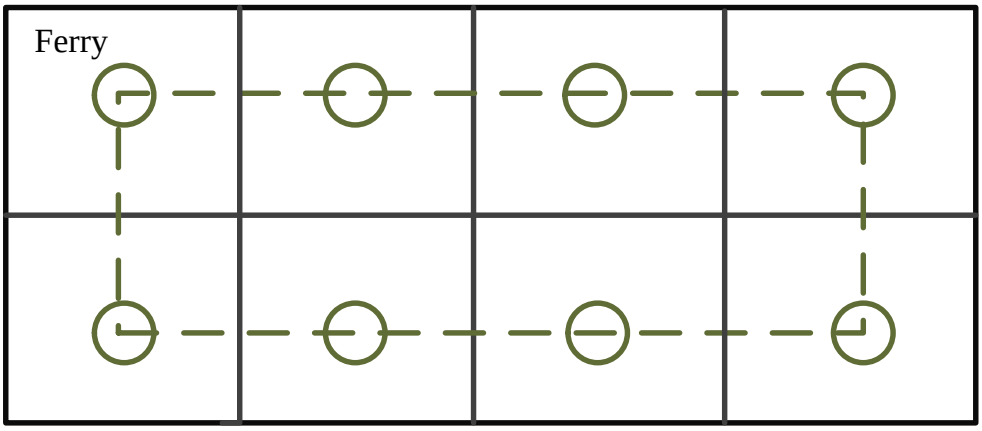


Figure 2.13: **Ferry Patrol Route [72]**

In [72], at each location that a ferry stops, the ferry will broadcast secret patrol service announcement messages. When the destination discovers that the ferry is

nearby and suspects its forwarding path is attacked, it will issue a service request message to the ferry. The ferry will provide the service to this destination. It will travel along the reverse path to probe the nodes along the forwarding path and find the malicious nodes. If the ferry finds a suspicious node who keeps dropping messages, it will define the node as culprit and broadcasts the news to other nodes.

Using the ferry idea, the authors proposed an improved version of MDS called FBIDM in paper [73]. When a node hears the service announcement of the ferry, it shares some encountering and delivering information with the ferry, helping the ferry distinguish the behavior of nodes. In the system, it requires that each node maintains a delivery predict ability to other nodes, storing the information in the delivery probability table (DPT). Besides, each node also maintains a delivery encounter table (DET). In the DET, each node maintains the last M values of delivery predictability to other nodes just before its connectivity with these nodes disappear and the time when this node loses such connectivity. By checking the history information from the DPT and DET, the ferry can validate the packet delivery probability of each node by cross-checking the reports between pairs of nodes. By incorporating the information from all nodes, the ferry can finally identify the potentially malicious nodes.

Considering the negative impact of high false positive rates in [73], [74] introduces the idea of transitive property to the system and proposes a MDS called MUTON to improve the effectivity of the system. Compared to the FBIDM MDS in [73], the MUTON MDS uses a self-examination approach: The ferry detects malicious nodes by only using the examining information stored at each individual node. Besides storing the table of DPT and DET (similar to FBIDM), each node also stores the latest delivery probabilities of other nodes that it encounters in the transitive information table (TIT). When the ferry broadcasts a service message to ask the node's information, the ferry only needs to examine the node itself based on the recorded information in its DPT, DET and TIT. By comparing the packet delivery probability calculated by the ferry with the claimed probability by the node, the ferry can determine whether the node is malicious.

However, all of the ferry detection methods need additional devices for examining the network which makes the system not economical and feasible. In addition, how to choose a suitable and fixed route for communicating with all other nodes is a big issue to address for the ferry in DTNs. Meanwhile, if an attacker is able to compromise the ferry node, the system is compromised therefore the result is untrusted anymore.

Some recent work has suggested the usage of encounter tickets to tackle the problem of misbehavior detection in DTNs [15, 16, 17]. The idea is that after a contact and transmission of data between two nodes, they provide each other with a ticket about this encounter. Upon subsequent encounters with other nodes these tickets will be exchanged as proof of a node's behavior in the past (see Figure 2.14).

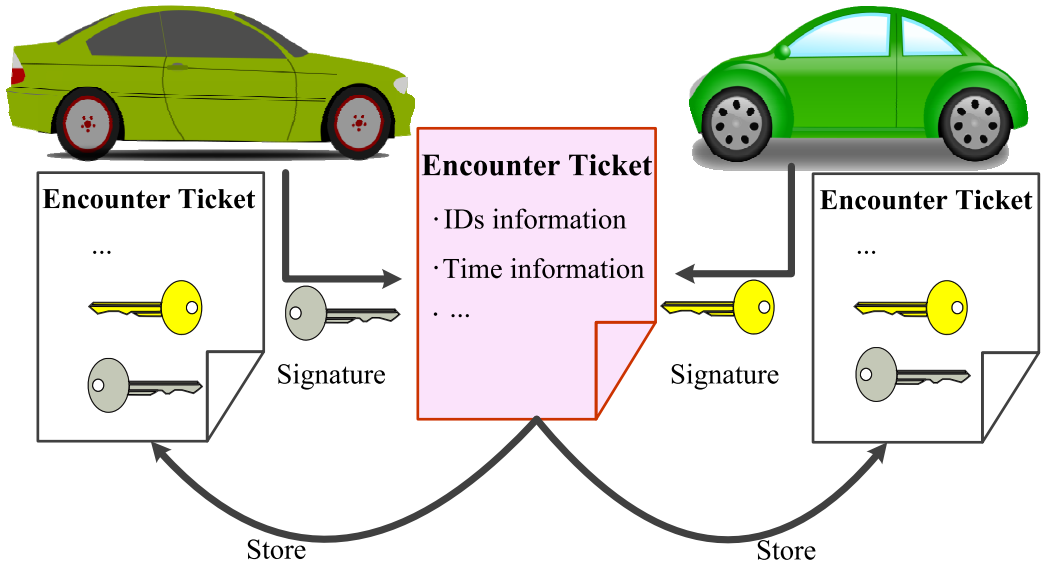


Figure 2.14: **Encounter Ticket**

In [15], the nodes depend on the probability of meeting a destination to evaluate an encounter's competency of delivering data. Encounter tickets are introduced here to secure the evidence of each contact. When two nodes encounter, they will first generate a mutually signed encounter ticket, certifying that they have been in contact at a certain time t. Afterwards, each node can use its encounter tickets to reveal its contact history to other future encountered nodes, proofing they are not blackhole nodes which exaggerate their ability to meet destinations. Although the encounter tickets can guarantee the correct information about the routing metric, a blackhole node can still launch the attacks and drop all the packets it receives using its true routing metric. However, the system does not provide the solution on counteracting that.

Similar to the idea of encounter tickets, [16] proposes a mechanism to detect blackhole nodes based on packet exchange recording. After any pair of nodes successfully exchange data, nodes will generate a packet exchange recording which includes the number of receiving and forwarding messages during this encounter and store this packet exchange recording in their memory. According to the exchanged message information in these packet exchange recordings, nodes can validate the behavior of an encountered node and determine whether the encountered node launches the blackhole attack. The authors also try to cope with the problem that the blackhole nodes cheat other nodes by dropping the packet exchange recordings from their memory. However, in their system, only after the same two nodes meet again, the node can verify whether the other encountered node drops the packet exchange recordings, making it not efficient for the network which exists dynamic network topology and

no continuous connectivity.

The usage of contact records to detect malicious nodes and mitigate blackhole attacks has been proposed in [17]. After two nodes met and exchanged messages, a contact record describing the encounter is created and mutually signed by both parties in DTNs. [17] extends the idea of encounter tickets, using contact records to record the information about the vector of packets buffered by nodes, the identifiers of the packets received by nodes and the identifiers of the packets sent by nodes. Based on the detailed information in these contact records, a node can detect if other nodes have dropped packets (see Figure 2.15). Since misbehaving nodes may misreport their contact records to avoid being detected, a node will disseminate the contact records from encountered nodes to a certain number of witness nodes and ask them for help with the decision-making process. Even though the system tries to prevent misreporting or collusion from the attackers by searching the help from the surrounding nodes, the system in [17] is still difficult to make a clear distinction for the packets which are dropped by the attackers or dropped because the buffer of nodes is full.

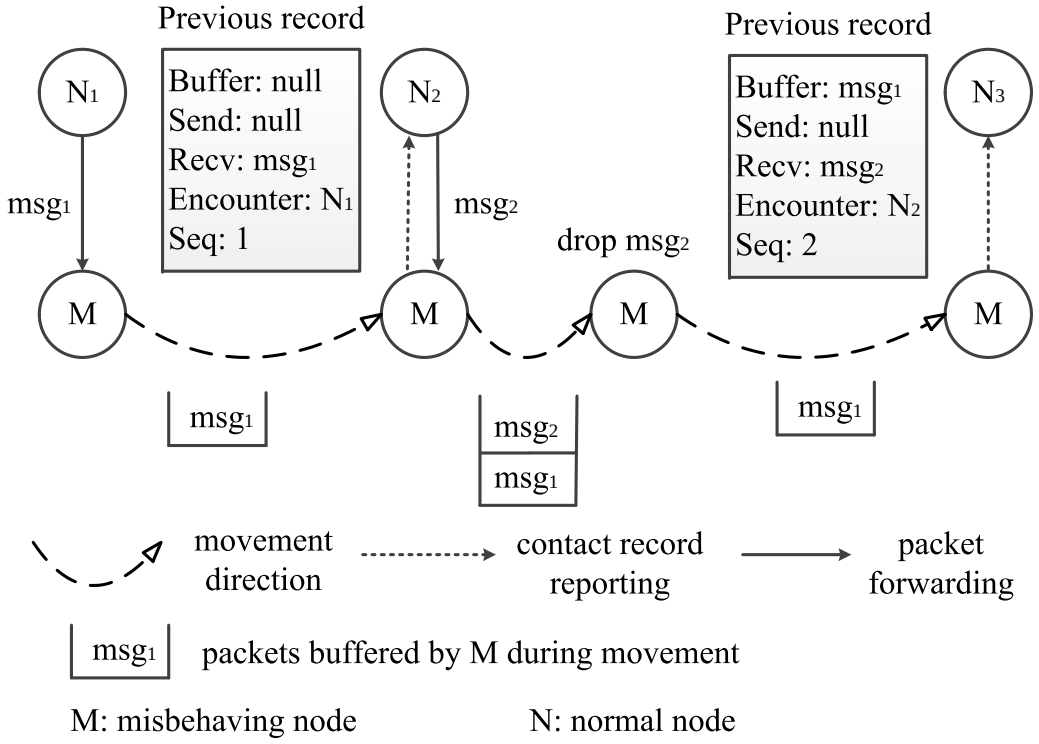


Figure 2.15: **Dropping Detection [17]**

However, the MDSs in [15, 16, 17] are designed to only detect blackhole attackers. Meanwhile, when attackers with the ability to forge encounter tickets exist, the system needs to rely on additional nodes in range to make a correct decision.

2.4.2 Credit Systems

The establishment of multi-hop wireless network relies on the assumption that nodes prefer forwarding packets for other nodes. Considering limited energy resources, if selfish nodes do not want to participate in the forwarding process, this may lead to the collapse of the network. One of the common ways to address the selfishness issue is using credit systems. The credit system introduces some form of virtual currency to regulate the packet-forwarding relationships among different nodes. By using credits, the system provides incentive to nodes for transmission.

In the credit system, the nodes (usually come from the source nodes or the destination nodes) need to pay the credits when they have packets to transmit. Each intermediate node can earn credits when they help other nodes forward their packets. In MANETs, papers [75, 76, 77] propose a credit-based approach based on a tamper-proof hardware to make sure that the correct amount of credits is deducted or added at each node.

Paper [75] presents a credit system to stimulate nodes to cooperate. Nodes providing a service to other nodes will be remunerated by using virtual currency "nuglet", while nodes receiving a service should be charged. The nuglets are paid in a one-to-one interaction fashion between nodes. The system implements two payment models, the Packet Purse Model and the Packet Trade Model. To implement the Packet Purse Model, before sending a packet, the source node should load some nuglets in the packet. Intermediate nodes will help forward the packet because they can acquire some nuglets from the packet if they cooperate. In the Packet Trade Model, each intermediate node purchases the data packet from its predecessor and sells it to its successor along the path. Eventually the destination pays for the received packet. A physically tamper-resistant module is responsible for all cryptographic operations related to the nuglet transfer. The tamper-resistant module makes sure that nodes can only obtain the corresponding nuglet when they cooperate and cannot cheat the system due to their greed.

If attackers flood the network with packets destined to non-existent nodes, especially in the Packet Trade Model, this will lead to the result that the nodes are unable to transfer the purchased packets and eventually lead the nodes to starvation status. Moreover, the per-hop payment for every packet incurs a high computational overhead. Besides, using the tamper-proof hardware increases the cost of the network and therefore makes the proposal not widely acceptable.

In MANETs, [78, 79, 80] propose a mechanism that instead of using the tamper-proof hardware, each node keeps its receipts of forwarding packets and uploads these receipts to a trusted centralized bank. Depending on the reported receipts from

the nodes, the bank determines the charges and credits for nodes involved in the transmission of packets.

In paper [78], because a route to the destination is already decided before sending packets in this system, the authors propose a simple, cheat-proof and credit-based system (Sprite) to motivate intermediate nodes to forward packets along this route. Sprite requires that every source node has to pay credits for sending its packets. Hence, in order to be able to send its packets, a node must earn credits by helping other nodes forward packets. In the system, nodes will be charged or rewarded credits based on "receipt". After receiving a packet and its corresponding receipt from other nodes, an intermediate node first stores the signed receipt of this packet, which was generated by the source node, and then forwards this packet to the next hop. Later when the node has a fast connection to the Credit Clearance Service (CCS), it uploads its collected and signed receipts. The CCS will reward the node based on the number of receipts and charge the corresponding sender. The CCS determines rewards and charges from a game theoretic perspective: A node can only receive credits when the next node on the path also reports a valid receipt to the CCS to acknowledge the successful transmission. Sprite also designs the corresponding mechanism to counter selfish behavior such as saving a receipt but not forwarding packets, falsely claiming the receipt without receiving the corresponding packet, or receiving a packet but not reporting the receipt.

However, in MANETs, the credit system is built on the assumption that there are end-to-end paths between the communication sources and destinations. Yet in DTNs, the network is highly dynamic and frequently disconnected, hence, it is unrealistic to maintain end-to-end paths between any communication source and destination pair. Hence, some researchers improve the credit system to adapt it to DTNs [81, 82, 83].

Paper [81] presents the Signature-Seeking Drive (SSD), a secure incentive framework for commercial advertisement dissemination in VDTNs. The system introduces virtual currency to regulate the packet forwarding, in other words, the data forwarding process can be also seen as data rewarding process. The virtual cashier (VC) is responsible for virtual currency issuing and clearance. SSD relies on the PKI to provide secure incentives for cooperative nodes.

According to the characteristics of advertisements (ads), local advertising or intensive advertising over a wide area, SSD proposes two strategies to disseminate ads, which are one-level and multi-level dissemination. For the one-level dissemination, a vehicular node first obtains an ad and a corresponding voucher about this ad from the ad distribution point (ADP). The voucher is tied with the vehicular node's identity and used to get the virtual currency for disseminating the ad later. Since becoming an ad-forwarding vehicle, the vehicular node will try to forward the ad to its neighbors and obtain receipts from these ad-receiving neighbors. After collecting enough receipts by forwarding the ad, the vehicular node will go to the VC and exchange

those collected receipts with virtual currency (see Figure 2.16). In the one-level dissemination model, both the ad-forwarding vehicles and the ad-receiving vehicles will get rewards. However, the ad-receiving vehicles will not keep transferring the ad, because they do not obtain the initial voucher of the ad and cannot get any rewards for doing that.

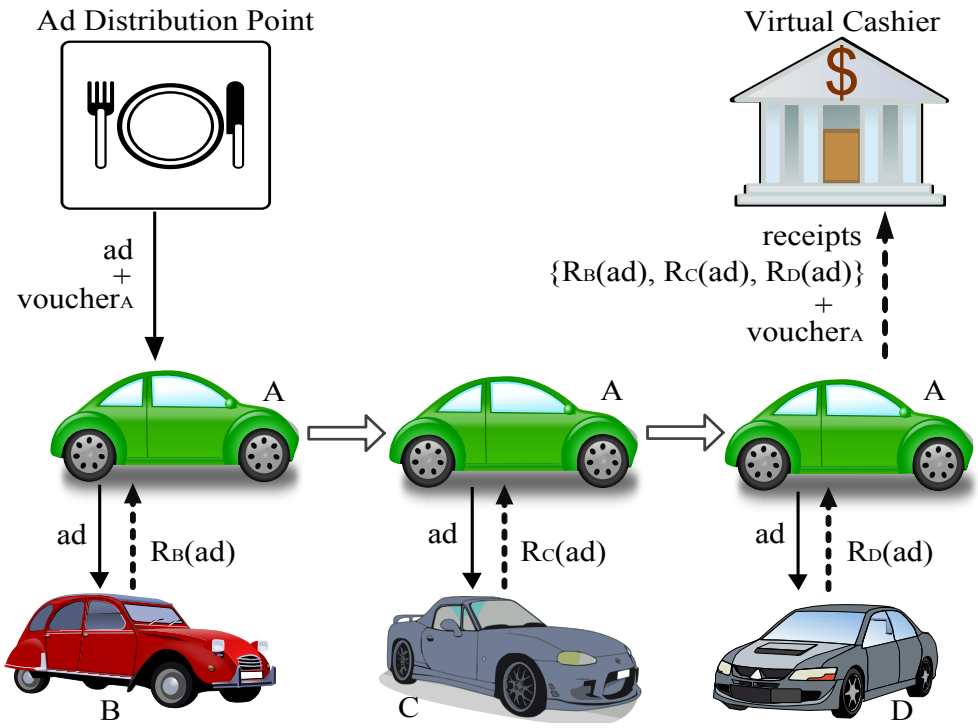


Figure 2.16: **One-level Ad Dissemination [81]**

To spray an ad in a wide area, the multi-level dissemination model is proposed here. The multi-level dissemination model allows an ad to keep forwarding by different nodes. However, this will increase a heavy outlay for the company if the ad is allowed to be unlimitedly transferred. Hence, the authors use the idea of "onion voucher" to improve the system to n-level dissemination model. Onion voucher requires only n nodes can forward this voucher. Meanwhile, when the voucher passes to the next ad-receiving nodes, all the corresponding vehicular nodes in each level need to sign the voucher with their signatures. And depending on the voucher and the collected receipts, the vehicular nodes can obtain the virtual currency from the VC.

SSD proposes a receipt counting reward scheme and gives vehicular nodes incentives for spraying messages. However, the system only targets to send the ad to as many nodes as possible and thus limited its applicability.

Secure multilayer credit-based incentive scheme (SMART) [82, 83] adopts a novel layer concatenation technique to resist cheating behavior of selfish nodes in DTNs.

SMART uses a virtual electronic currency, called a layered coin, to stimulate the cooperation among nodes.

A layered coin is composed of the base layer and the endorsed layers (see Figure 2.17). The source node generates the base layer which includes the packet and its related information, such as reward policies and the class-of-service (CoS) requirement. By checking the base layer, each intermediate node will know the predefined CoS requirement and the reward policy of the packet. Each intermediate node generates the endorsed layer based on the previous layers by appending its own unforgeable digital signature. Using the layer concatenation technique, it is easy in SMART to track the propagation path and determine the forwarding nodes by checking the signature of each endorsed layer. When the packets are finally forwarded to the destination node, each intermediate node can share the credit defined in this coin. When the packet fails to transfer to the destination node, SMART will give good reputation values to these intermediate forwarding nodes which cannot get credit from the coin.

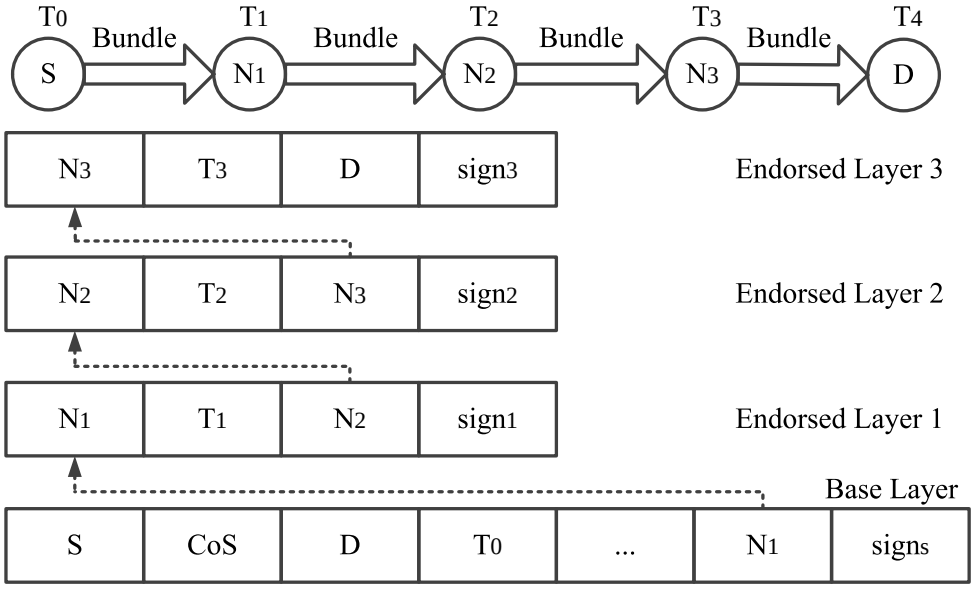


Figure 2.17: **Layered Coin Architecture [82]**

A layer concatenation technique ensures the security of layered coins, however, it increases the computation and transmission overhead in the system, making it unaffordable in some DTN applications. Besides, most of the credit systems only focus on the problem of selfish nodes, if blackhole and greyhole attackers exist in the system, it is difficult for the credit system to cope with that.

In VDTNs, considering energy and memory, without giving incentives, vehicular nodes may be reluctant to cooperate if it is not directly beneficial to them. Therefore, our system will utilize the idea of the credit system to encourage vehicular nodes forwarding other nodes' messages.

2.4.3 Tit-for-Tat Systems

Tit-for-Tat (TFT) is a simple, robust and practical strategy that links incentive mechanisms with reputation: A node will collaborate and provide good service to a cooperative neighbor. Meanwhile, the node will autonomously lower the service or even isolate a neighbor if it detects that the neighbor is misbehaving.

TFT [84] is first proposed in Robert Axelrod's tournament. TFT has won both of the computer tournaments because of its remarkable strategy: TFT cooperates in the first round and thereafter always does whatever the counterpart has done in the previous round. It will cooperate if the counterpart has cooperated, and it will defect if the counterpart has defected. BitTorrent is a successful example to employ TFT for overcoming free-riding behavior in peer-to-peer (P2P) file sharing systems [85, 86, 87]. TFT ensures that only nodes who actively contribute are allowed to download files from others. Much work borrows the core idea from the TFT strategy and applies it widely in wireless ad hoc networks [88, 89, 90].

The paper [88] describes a game-theoretic algorithm, based on the Generous Tit-for-Tat (GTFT), to optimize throughput performance for all nodes in the network. In their work, nodes are classified in different energy classes. Depending on their current energy classes, nodes decide whether they will forward packets for their neighbors. The packet forwarding decision can be modeled as a repeated Prisoner's Dilemma game, nodes employ the GTFT to achieve an optimal Nash equilibrium. When nodes rely on other nodes to help them transmit packets, there is a mutual dependency relationship between nodes. Their cooperative behavior can encourage cooperation among nodes, while their uncooperative behavior can lead to refused service from others. Using the GTFT as the foundation for the system, eventually nodes can achieve the optimal trade-off between throughput and lifetime.

However, in paper [88], the authors do not take the network topology into account. In addition, the approach only focuses on solving selfish node problem but cannot cope with other type of misbehaving nodes.

In DTNs, considering the selfish behavior of nodes, [91] uses the pair-wise TFT as a robust and practical incentive mechanism to stimulate cooperations among nodes. Meanwhile, [91] proposes an incentive-aware routing protocol that improves the system-wide performance and allows selfish nodes to maximize their individual utility while conforming to TFT constraints.

The proposed TFT-based algorithm focuses on detecting good behavior. Destination nodes disseminate packet acknowledgements as the verification of successful relaying. These acknowledgements help to encourage all the intermediate nodes by increasing their TFT counters. Moreover, the TFT-based algorithm incorporates

generosity and contrition to address the bootstrap and link variation problem. Bootstrap is a problem for TFT, since at the start, when there is no successfully relayed packets between nodes, traditional TFT will prevent the communication between these nodes. Generosity is introduced in the paper to solve this problem. When two nodes encounter for the first time, generosity allows a node to send ε number of packets more than it has earned, which stimulates the beginning of cooperation. Once any imbalance exceeding ε occurs, this will lead to lengthy retaliation between two encountered nodes. Hence, contrition is proposed to prevent mistakes from causing endless retaliation. In this paper, contrition is only responsible for providing a way to return to stability after perturbation, but provides no way to reach that stability.

However, selfish nodes can exploit the generosity from other nodes and reject to provide the generosity in return. Hence, this TFT-based method cannot fully avoid selfish behavior.

In this thesis, we adopt the TFT strategy to improve the efficiency of our proposed MDS in section 5.2.3.

2.5 Conclusions

In this chapter, we have discussed some related work, including the recent VDTN applications, the common DTN routing protocols, the security defense systems using the idea of encryption and authentication, and the MDSs aiming of detecting malicious nodes. Since none of the existing MDSs can fulfil all requirements and completely address security problems in VDTNs, in the following chapters, we will present the architecture of our MDS in detail and address malicious nodes in VDTNs.

Chapter 3

Basic Architecture of Misbehavior Detection System

3.1 Introduction

In vehicular networks, vehicular nodes are designed to communicate wirelessly to help other nodes transfer messages. However, the unique network characteristics, such as lack of contemporaneous path, high variation in network conditions and difficulty to predict mobility patterns, bring many issues to design such a network. DTNs are a suitable technology to solve these problems for vehicular networks. Even though we establish a VDTN, we cannot assume that all nodes are altruistic in such a network. Especially if multiple attackers exist and collude to disrupt the network, this can cause severe problems in VDTNs.

Some work in this area has focused on authentication and encryption as we introduced it in section 2.3. However, authentication and encryption cannot prevent the legitimate nodes from launching attacks to the network. Additionally, the characteristics of a VDTN make most traditional strategies of detecting attackers infeasible. A traditional MDS approach prefers to observe nodes' data forwarding behavior to make a judgement. However, because of the "store, carry and forward" paradigm in VDTNs, it is difficult to observe data forwarding directly in a network where the end-to-end connectivity can not be guaranteed. Hence, an efficient MDS to detect malicious nodes in VDTNs is imperative.

In this paper, we propose a general mechanism that detects selfish nodes, faulty nodes, blackhole attackers and greyhole attackers with varying drop probabilities (see section 3.2.2). Depending on the amount of information available when two nodes meet, the system will adaptively choose its detection parameters to maximize detection rates while minimizing false positive rates. Furthermore, we introduce a trust reputation system to encourage cooperation. Nodes with a good trust reputation will

exchange messages with each other, and may even share information about the detected malicious nodes while nodes with a very low trust reputation will be excluded from the network.

3.2 System Model

In this section, we first introduce a framework that proposes a guideline for the vehicular node model. Then, we illustrate the different misbehaving activities of attackers in the system. Unless otherwise mentioned, we assume that all nodes represent vehicular nodes in our system. All nodes need to store information about prior interactions with other nodes. The data structures kept by the vehicular nodes are defined in the *Vehicular Node Model*. Further, we assume that there are benign as well as malicious nodes. The malicious nodes follow an *Attack Model*.

3.2.1 Vehicular Node Model

In this thesis, normal vehicular nodes will follow the *Vehicular Node Model*. They utilize the resources provided by other nodes and meanwhile provide their service to other nodes in vehicular networks.

We assume that each vehicular node possesses an unique identifier. Each vehicular node could obtain its own private and public key pair by using the IBC (see section 2.3.2). Generally speaking, a vehicular node's private key is only known by itself and other nodes cannot forge it. Furthermore, we require the network to be loosely time synchronized: This condition is easy to fulfill, as it is to be expected that all VDTN-enabled vehicles will be equipped with a GPS (Global Positioning System) receiver for navigation purposes that can also provide a very precise clock reference. After successfully exchanging messages with another vehicular node, both nodes will generate an *Encounter Record* (ER) and store it in their memory. Afterwards, the ERs will be exchanged upon each contact. Because the ERs provide some provable information about a node's behavior in the past, the exchanged and collected ERs are used to assess and weigh the behavior of nodes against each other. Additionally, each node will store three lists in its memory: the *Meeting List* (ML), the *Local Blacklist* (LBL) and the *Friend Blacklist* (FBL). The ML stores the information related to previously encountered vehicular nodes. All the detected malicious nodes will be put in the LBL. And the FBL is used for collecting the information from the trusted friend nodes. The detail information of these lists will be described in the following part.

3.2.1.1 Encounter Record

The basic idea of ER is to generate an unforgeable evidence to show the history of the vehicular nodes. An ER will be generated after two vehicular nodes met and successfully exchanged messages. Then the ERs will be stored in the node's memory as a proof documenting the node's behavior during previous encounters. When a vehicular node meets another vehicular node, before transferring messages, both nodes first need to submit their own ERs for verification. By checking the history of an encountering node, the node can make a judgement about the behavior of the encountering node and decide whether to help it transfer the messages.

In our system, the node who starts the communication will generate a new ER to record this encounter, use its private key to sign the ER and then send the ER to its encountering node. Afterwards, the encountering node will check the content of the received ER, sign the ER with its private key if the content is correct, send the signed ER back to the node and keep a copy in its memory. The basic process of the ER exchange is shown in Figure 3.1. Here we use node i and node j as an example to illustrate how the ER is constructed. Node i generates the ER for node j as follows:

$$\begin{aligned} ER &= ID_i,\ ID_j,\ sn_i,\ sn_j,\ t,\ Re_{i\to j},\ Re_{j\to i} \\ Re_{i\to j} &= \{(msg_{id},\ msg_{src} \mid i\ sent\ msg\ to\ j)\} \\ Re_{j\to i} &= \{(msg_{id},\ msg_{src} \mid j\ sent\ msg\ to\ i)\} \\ \sigma_i &= sign_{SK_i}\{H(ER)\} \\ \sigma_j &= sign_{SK_j}\{H(ER)\} \\ ER^* &= ER,\ \sigma_i,\ \sigma_j \end{aligned} \tag{3.1}$$

1) The ER includes both of the nodes' identifiers, ID_i and ID_j. The *ID* of each vehicular node is unique. In a realistic world, the license plate may use as the *ID* of vehicular nodes, some researchers have worked on this area [6, 63, 67, 92, 93].

2) Each node possesses its own unique sequence number *sn* that starts from a fixed number and is increased after each contact. Vehicular nodes must comply with two basic principles for *sn*: First the vehicular node is not allowed to use the same *sn* twice; secondly the sequence numbers applying in the vehicular node's ERs should be sequential.

3) t indicates the time when this ER was generated. In our system, the (sn, t) combination in the ER holds that a greater *sn* for a given *ID* also implies a greater t. The (sn, t) information can be used to detect most basic manipulations such as

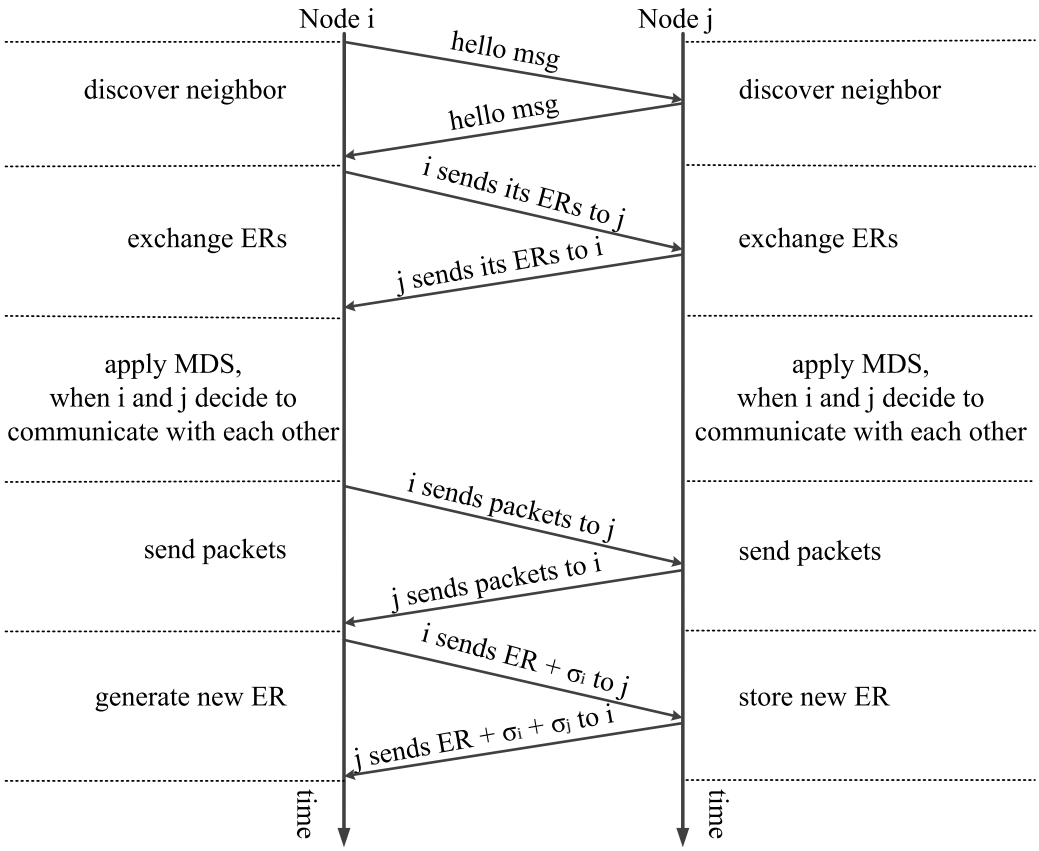


Figure 3.1: **The Process of Encounter Record Exchange**

omitting or dropping ERs. To prevent the overflow problem, we set sn and t as fixed length fields in our ER.

4) In paper [16], the ERs record the number of receiving packets and forwarding packets in each contact and then the ERs use this information to detect blackhole attackers. When the behavior of vehicular nodes is not homogeneous, the false positive rate of the system will become very high. The ERs in [17] include the information about a vector of packets buffered by nodes, the identifiers of the packets received by nodes and the identifiers of the packets sent by nodes. These ERs are also used to detect blackhole attackers. However, the system in [17] is difficult to make a clear distinction between the packets which are dropped by the attackers or dropped because the buffers of nodes are full. Additionally, both of the MDSs in [16, 17] rely on specific routings, which makes them not widely applicable in different scenarios. In our system we introduce the Re set, which identifies the transmitted messages. The $Re_{i \to j}$ set consists of 2-tuples (id, src) storing for each message that has been sent from i to j, the message's id and the id of the originating node. $Re_{j \to i}$ contains the message information sent from vehicular node j to i. By obtaining the information from the Re set, we can calculate different indicators from the ERs to characterize

a node's behavior and propose different mechanisms to build the MDS. Meanwhile, using the *Re* set, our proposed MDS can be applied through different DTN routing protocols.

5) Both communication partners need to cryptographically sign an ER (σ_i, σ_j) to ensure its authenticity and integrity. $sign_{SK_i}\{*\}$ and $sign_{SK_j}\{*\}$ denote the signature using node i and j's private key. Here we use $H(*)$ to denote a hash function. Figure 3.2 shows how the nodes mutually sign the ER.

6) Considering the memory of vehicular nodes, we define that each node can store the maximum number of w ERs. Once enough ERs have been introduced into nodes' memory, we will use first-in-first-out (FIFO) policy to deal with older ERs.

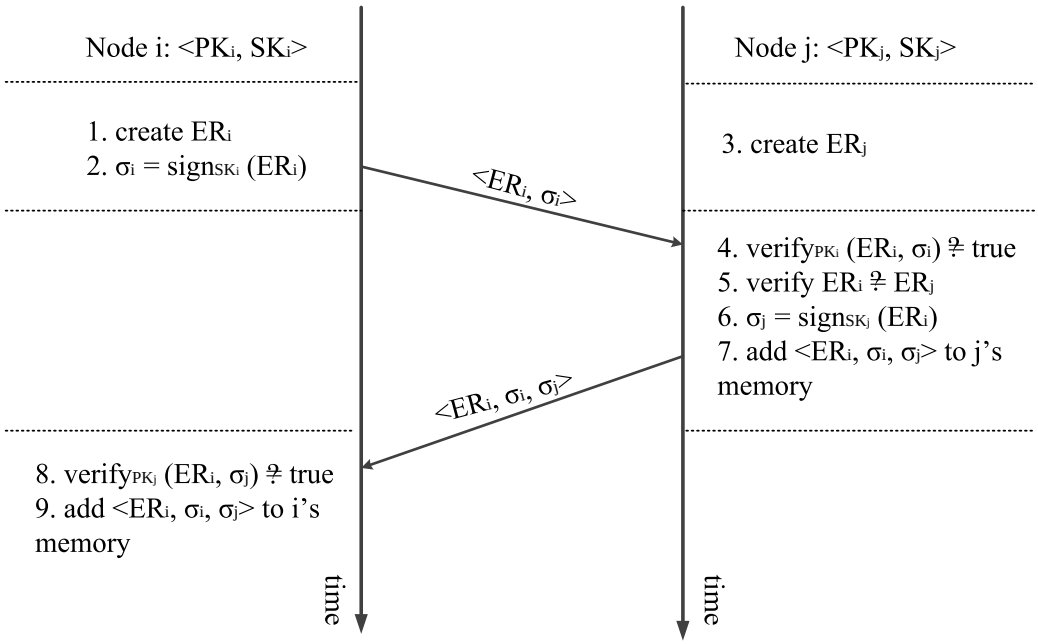


Figure 3.2: **The Process of Signing Encounter Record**

3.2.1.2 Meeting List

In the ML, a node stores the information from previously encountered vehicles. When an encountered node is not known by this node before, a new entry about the encountered node will be created in the ML. If the node meets an encountered node more than once, the corresponding newer entries about the encountered node will be updated.

The detail of the entries in the ML is as follows: Each record in the ML (show in Table 3.1) includes the information of the identifier *ID* of the encountered node, the

unique sequence number *sn* of the encountered node, the time *t* of the contact and the Trust Reputation (TR) (see section 3.3.2.1) assigned to the encountered vehicular node. The *option* entry in the ML is used to extend the other useful information about the encountered nodes. We use the ML entries to check the validity of ERs later, as they store the most recent known (*sn*, *t*) combination for a vehicular node: In a normally operating network without any forged ERs, the condition holds that a greater *sn* for a given *ID* also implies a greater *t*. This relation will be used by the *Rule Violation Checks* (see section 3.3.1) when checking new ERs. In Chapter 5, we use the *option* entry in the ML to store the *Message Forwarding Ratio* θ and the *Message Receiving Ratio* ψ of encountered vehicular nodes (see section 5.2.1). The θ and ψ are the main parameters when our MDS uses the cluster analysis to define the behavior of encountered nodes.

Table 3.1: **Format of the Meeting List**

ID	***sn***	***t***	**TR**	***option***
Node 1	1101	10000 s	0.80	(1.10, 1.0)
Node 2	1010	9000 s	0.28	(0.40, 2.0)
Node 3	1118	10050 s	0.26	(0.20, 3.0)
Node 4	1508	10210 s	0.70	(0.85, 1.0)
Node 5	1008	8050 s	0.82	(1.20, 1.0)
Node 6	1098	9901 s	0.90	(1.70, 1.0)
Node 7	1883	19005 s	0.72	(0.90, 1.0)
...	...	...	...	...
Node n	2018	20050 s	0.60	(0.80, 1.0)

3.2.1.3 Local Blacklist

All malicious nodes locally detected by a vehicular node will be put into that vehicular node's LBL. A vehicular node will refuse to transmit messages to or receive messages from vehicular nodes in its LBL. The entries in the LBL (show in Table 3.2) includes the information of the identifier *ID* of the detected node and the detected time *t*. To encourage more vehicular nodes to participate in data forwarding in the network, we provide an incentive mechanism for misbehaving vehicular nodes: All records in the LBL have an expiration time. After a predetermined time, an entry in the LBL expires and the previously detected node is allowed back into the network. However, if a node from the LBL is allowed back to the network, it is on "probation", meaning it will start with a lower initial TR than other nodes that have just joined the network. If a node is removed from the LBL and assigned a new initial TR, except the TR entry, other current entries for that node in the ML will not be changed.

Table 3.2: **Format of the Local Blacklist**

ID	***t***
Node 2	9000 s
Node 3	10050 s
...	...

3.2.1.4 Friend Blacklist

In our system, the function of the FBL is optional: Vehicular nodes can make an independent decision whether they want to apply the function of the FBL. When two nodes both hold a high TR in their ML for each other, these nodes will become friend nodes. When nodes do not apply the function of the FBL, these friend nodes will not share information. However, when nodes are friends and want to apply the FBL, they will accept each other's LBL records which are not in their own LBL and add them to their FBLs. Based on trust, vehicular nodes will only transmit their own LBL but not their FBL. Each record in the FBL (show in Table 3.3) includes the identifier *ID* of the accused vehicular node, the detected time t which provides by its first accusing vehicular node and a group of *ID*s of nodes, which have accused this suspect node. Only when a vehicular node gets enough evidence from different friends, it will consider the suspect node as an evil node and refrain from exchanging messages with it.

Table 3.3: **Format of the Friend Blacklist**

ID	***t***	**Group of accusing nodes' *ID***
Node 20	11000 s	(Node 1, Node 5, Node 6)
Node 50	12000 s	(Node 1, Node 5)
...	...	...

3.2.2 Attack Model

Since vehicular nodes in VDTNs are owned and controlled by rational entities, such as people or organizations, these nodes usually behave differently. In our system, besides the normal vehicular nodes which transfer messages for others and also get help from other nodes as we described in section 3.2.1, we consider three kinds of VDTN nodes to cope with: Selfish nodes, faulty nodes and malicious nodes. These nodes will follow the *Attack Model*.

Vehicular nodes are controlled by rational users, some of them will perform according to their selfish nature [83, 94, 95, 96]. Considering that forwarding messages for others in VDTNs consumes their own energy and other resources, selfish nodes are not willing to serve for others. However, they attempt to maximize their own utilities and conserve their limited resources for their own messages. Since most routings rely on an inherently cooperative activity of nodes, such selfish behavior could significantly degrade network performance.

Due to the hard- or software failures of a vehicular node, some of the faulty nodes may not send or forward messages, or some of the faulty nodes cannot work according to the principle which the normal nodes should obey. Because of the scarce transmission opportunities in VDTNs, the system should detect these faulty nodes as fast as possible to avoid waste of relaying opportunities and prevent the loss of packets [64, 68, 97].

In VDTNs, malicious nodes try to attack the system with the intention of disrupting some part or even the whole network. Malicious nodes may launch the attacks to the network: They can modify their forwarding history to assert a good path through them, attract messages and finally drop messages instead of forwarding them. The most common attacks are blackhole and greyhole attacks. Blackhole attackers [15, 98, 99, 100, 101] will attract messages and then drop all messages they receive instead of forwarding them. When only partial droppings occur, the attacker is referred to as a greyhole attacker, which is much more difficult to detect [72, 102, 103, 104]. In this thesis, we use a parameter called drop probability to describe the behavior of greyhole attackers. If a given node's drop probability is $p = 1$, it is a blackhole attacker. Drop probabilities $0 < p < 1$ characterize a greyhole attacker. For example, if the drop probability of a node is $p = 0.6$, it denotes that the node will have 60% probability to drop the messages after it obtains the messages. A bigger drop probability denotes a higher probability that the node will drop the received messages. Blackhole and greyhole attackers can significantly impact the network performance, therefore, they need to be excluded from the network as fast as possible. Additionally, if a normal node's communication system is penetrated by attackers it might become malicious and drop messages.

Considering energy and memory, without giving incentives, selfish nodes may be reluctant to cooperate if it is not directly beneficial to them. And the system should detect faulty nodes, exclude them from the network and notify the operator. Moreover, the nodes want to detect blackhole and greyhole attackers as fast as possible to avoid wasting resources by relaying to uncooperative nodes. Therefore, our MDS will focus on detecting and avoiding blackhole and greyhole attackers, which may either be the result of a deliberate attack or the result of hard- or software failures, while encouraging selfish nodes to forward other nodes' messages.

3.2.2.1 Basic Attacks

Basic attacks are the behavior when nodes will independently try to disrupt the network. A selfish node utilizes network resources but does not cooperate with other nodes, saving the battery for its own communication. This kind of behavior does not directly damage other nodes, but disturbs the effectiveness of the network. A faulty node may have the problem of sending or forwarding messages or generating wrong ERs. By dropping messages, blackhole and greyhole attackers destroy the network, consuming the available energy and occupying valuable bandwidth. The classification of behavior is described as follows:

1) *Behavior 1*: A selfish node will try to receive or forward less messages which are generated for other nodes.

2) *Behavior 2*: A faulty node cannot send or forward messages, or cannot generate correct ERs.

3) *Behavior 3*: A malicious node will first receive messages, and then according to its drop probability, decides whether to **drop the messages**. If a given node's drop probability is 1 ($p = 1$), it is a blackhole attacker. Drop probabilities $0 < p < 1$ characterize a greyhole attacker.

3.2.2.2 Advanced Attacks

Attackers, which come from selfish nodes or malicious nodes, use advanced attacks to conceal their malicious behavior. By partially hiding their history or colluding with other attackers, they try to deceive other vehicular nodes. Their behavior is described as follows:

4) *Behavior 4*: It is not practical to store and exchange all ERs, hence, in our system a normal node only stores w new and sequential ERs in its memory to provide them to other nodes for verification. However, a selfish or malicious node will store as many beneficial ERs for itself as it can and **randomly choose** these **better ERs** from its memory to present them to others.

5) *Behavior 5*: To get good ERs and maintain the sequence of its *sn*, a selfish or malicious node can **use the same *sn*** to generate ERs **with different nodes** until the content of the ER is beneficial for it. Afterwards, it updates the *sn* to generate new ERs.

6) *Behavior 6*: A selfish or malicious node generates a group of sequential and good ERs by adhering to the rules. After it accumulates enough good ERs it commences the attack, but **does not store any new ERs generated during the attack**

phase. After a while it may decide to generate a new set of sequential and good ERs by abiding the rules of the network.

7) *Behavior 7*: If a selfish or malicious node provides all of its ERs to the encountering nodes, it shows its whole history to others and this may increase the chance to be detected by others. Hence, a selfish or malicious node can **drop some of its older ERs** and only provide few new and sequential ERs for other nodes to detect its behavior.

8) *Behavior 8*: During one contact, a selfish or malicious node purposefully **transfers a huge number of messages** to the encountered node, making the generated ER influence the result of the ER group to conceal its bad behavior.

9) *Behavior 9*: When these attackers collude with each other, they forge beneficial ERs for themselves. We assume that attackers can use two ways to collude. One way is that when two attackers are in each other's communication range, they will generate a bogus ER describing a non-existing contact between them and both sign for the ER.

10) *Behavior 10*: The other way is that the attackers join in **a colluding group**. For a colluding group of attackers it is reasonable that the colluding nodes will generate bogus and correctly signed ERs. Hence, we assume malicious nodes know each other's private key allowing them to **forge ERs**: In our system, when the attackers want to improve their ER results, there is a 50% chance for a malicious node to independently generate a bogus ER describing a non-existing contact between the colluding peers, which come from its colluding group.

3.3 System Architecture

The presented MDS can be decomposed into three main components: The *Rule Violation Checks*, the *Evaluation Module* and the *Decision Module*. Our MDS uses the *Rule Violation Checks* to assert the validity of the ERs provided by the encountered nodes. Then using the ERs, the *Evaluation Module* can assert the trustworthiness of other nodes, assign and update the TR (section 3.3.2.1) to other nodes. Based on the TR, nodes will be treated differently by the *Decision Module*.

3.3.1 Rule Violation Checks

When vehicular nodes meet each other, first ERs will be exchanged. To check the behavior of the encountering node, the node will first use the information in the ERs of the encountering node and its ML to check for any obvious violations of the constraints laid out in section 3.2.2.2. This procedure is executed by the *Rule Violation Checks*. The *Rule Violation Checks* module is responsible for detecting the behavior which aims at deceiving our MDS. The behavior includes **randomly presenting good ERs**, **using the same *sn* multiple times** until a good record is created, **not storing ERs** which exposes its malicious behavior or **dropping of older ERs** to provide less information trying to mitigate the risk of detection. After the *Rule Violation Checks*, only the usable ERs will be applied in the *Evaluation Module*. Additionally, with the help of the *Rule Violation Checks*, the node can know more knowledge of the whole network.

In the *Rule Violation Checks*, vehicular nodes must comply with two basic principles: First the sequence number sn for a node in its ERs should be sequential, and secondly it must hold that a greater sn for a given node also implies a greater t. If a node belongs to the faulty node class that cannot generate correct ERs or a node tampers with its ERs using the behavior as we described in section 3.2.2.2, our MDS can detect it immediately. Here we use node i and node j as an example to illustrate how the *Rule Violation Checks* works (see Algorithm 1). Node i checks the ERs from node j as follows:

1) If node j is in i's LBL, node i will refuse to transfer to and receive messages from j and the following checks will be omitted. Or if the node uses the function of the FBL (section 3.3.3.1) and enough friend nodes accused vehicular node j according to the FBL, the following checks will also be omitted (see Algorithm 1 lines 1-4).

2) If the sequence numbers for j in the ERs provided by j are not sequential, j will be put into the LBL immediately (lines 5-11).

3) If vehicular node j hides some random bad ERs, the remaining ERs are not sequential anymore, and thus j will fail the previous check. Therefore, node j might try to omit the newest ERs. This will not affect the sequentiality, however, it can be detected if the ML contains a higher sn_j for node j than the highest sn_j that can be found in the ERs provided by j (lines 12 and 29-33). Alternatively vehicular node j might try to drop older ERs which leads to less than w ERs provided by it. However, when a good vehicular node provides less than w ERs, it means that the node transmitted all ERs available to it. Thus, malicious dropping of older ERs can be detected, if vehicular node j provides less than w ERs, but the corresponding sn_j from the ML is *not* included in the ERs provided by j (lines 20-22 and 34-38).

Algorithm 1: **Check for Rule Violations**

ER_j is the list of ERs received from j

```
1: if (j is in LBL or FBL.numAccuser(j) ≥ M) then
2:     REJECT(j)                                   ▷ Do not communicate with j
3:     break
4: end if
5: for n = 1 to ER_j.length do
6:     if ER_j[n − 1].sn_j ≠ ER_j[n].sn_j − 1 then
7:         LBL.ADD(j)
8:         REJECT(j)                               ▷ j's sn is not sequential
9:         break
10:    end if
11: end for
12: maxKnownSn = ML.GETSN(j)
13: mlSNfoundInER = false
14: t = ML.GETT(j)
15: maxSNinER = 0
16: for all ER in ER_j do
17:    if ER.sn_j > maxSNinER then
18:        maxSNinER = ER.sn_j
19:    end if
20:    if ER.sn_j == maxKnownSn then
21:        mlSNfoundInER = true
22:    end if
23:    if ER.sn_j > maxKnownSn and ER.t < t then
24:        LBL.ADD(j)
25:        REJECT(j)                               ▷ j's (sn, t) tuples are inconsistent
26:        break
27:    end if
28: end for
29: if maxKnownSn > maxSNinER then
30:    LBL.ADD(j)
31:    REJECT(j)                                   ▷ j is hiding recent ERs
32:    break
33: end if
34: if ER_j.length < w and not mlSNfoundInER then
35:    LBL.ADD(j)
36:    REJECT(j)                                   ▷ j is hiding old ERs
37:    break
38: end if
39:                                                ▷ Update ML
40: for all ER in ER_j do
41:    if not ML.CONTAINS(ER.ID_x) then            ▷ x is j's encountered vehicular node
42:        ML.ADD(ER.ID_x, ER.sn_x, ER.t)
43:    else if ER.sn_x > ML.GETSN(ER.ID_x)
           and ER.t < ML.GETT(ER.ID_x) then
44:        LBL.ADD(ER.ID_x)                        ▷ x's sn is inconsistent
45:        ER_j.REMOVE(ER)
46:    else
47:        ML.UPDATE(ER.ID_x, ER.sn_x, ER.t)
48:    end if
49: end for
50: UPDATETR(j, ER_j)
```

4) Next the *Rule Violation Checks* will ensure that the j's sn and t fields from the ERs are consistent with the entries in the ML. For every sn that is larger than the sn for j in the ML it must hold, that the time t from the ER is also larger than the

t recorded in the ML for vehicular node j. If an inconsistency is detected, vehicular node i will add j to its LBL (lines 23-27).

5) Finally, the ML will be updated. If the information about an encountered vehicular node from j's ERs is not known before, a new entry will be created in the ML. If j's ERs contain newer sn and t records for a vehicular node already in the ML, the corresponding ML entries will be updated. However, if vehicular node i detects contradictions between sn and t of an ER and the ML for a vehicular node, it will add that node to its LBL and the corresponding ERs will not be used for the following checks (lines 40-49). Integrating information from another node's ERs into their ML allows vehicles to get information about nodes that they never met by themselves. This means updated knowledge is propagated quickly through the network.

If a node fails one of the basic rule checks in the *Rule Violation Checks*, it will be put in the LBL immediately and no communication with that node will take place. If node j passes the checks of Algorithm 1, the *Evaluation Module* will be used to update the TR of j. Good behavior will be encouraged, malicious or selfish behavior will be punished.

3.3.2 Evaluation Module

In the *Evaluation Module*, vehicular nodes measure the trustworthiness of other vehicular nodes, using TR. Based on the exchanged and collected ERs, the system changes a node's TR to assess and weigh the behavior of nodes against each other. In the following section, we introduce the detail of TR and explain the TR update principle.

3.3.2.1 Trust Reputation

In the *Evaluation Module*, vehicular nodes determine and update the TR of other nodes. The TR is a rating that vehicular nodes use to evaluate the trustworthiness of other vehicular nodes according to their precious activities. In our MDS, we use a decentralized way to assign the TR, that each vehicular node maintains the TR of other nodes according to its own judgement. As we mentioned before, the ML (see section 3.2.1.2) consists of the information about nodes' TR. In our system, we configure the same initial TR ($T_{initial}$) for all nodes which first join in the network. During the procedure of detecting other encountered nodes, when there is sufficient evidence accusing an encountered node, the TR of the encountered node will be decreased by the vehicular node. In our MDS, vehicular nodes assign different weights to different types of malicious behavior, namely a greater weight for significantly malicious behavior, a smaller weight for unsurely malicious behavior. Meanwhile,

benign behavior of an encountered node will be encouraged hence the vehicular node will increase the TR for it. The weight increasing mechanism follows the same way as the weight decreasing mechanism. Based on the *Re* sets in the ERs provided by the encountered nodes, different strategies will be applied to calculate and update the TR of the encountered nodes (see section 4.2 and 5.2).

When the vehicular node obtains the TR of the encountered node, thresholds are used to decide whether a given TR value denotes a normal or malicious node. Here we use the evil threshold T_{evil} and the friend threshold T_{friend} to classify evil and friend nodes. If the TR of a node in the ML has deteriorated so much as to fall out of the T_{evil}, it indicates that the encountered node is a malicious node. A TR between the T_{evil} and the T_{friend} identifies a normal node. When the TR of an encountered node is above the T_{friend}, it shows the encountered node is a friend node. These thresholds are static, however, the update of the TR will be subject to the algorithm we establish.

In our system, the TR is a measure for the trustworthiness of a node and can range between 0 and 1. Every node assumes an initial TR of 0.5 ($T_{initial} = 0.5$) for nodes it meets for the first time. The T_{evil} and the T_{friend} is equal to 0.3 and 0.8 respectively ($T_{evil} = 0.3$, $T_{friend} = 0.8$). We also consider the condition when a vehicular node is once detected as an evil node and after a while is allowed to communicate with other nodes again. If an entry in a node's LBL expires after a predetermined time, the owner of the LBL will allow exchanging messages with the previously detected node again, however, the former offender will be assigned an initial TR of 0.4 (see section 3.2.1.3).

3.3.2.2 TR Update Principle

Different behavior of a node leads to different trustworthiness of this node. Benign behavior of a node will be rewarded, therefore, the TR of the node will increase. Likewise, malicious behavior of a node will be punished and thus the TR of the node will decrease. As mentioned before, we have given the definition of the T_{friend} and the T_{evil}. When a node's TR value exceeds the T_{friend}, it shows the node keeps helping others and should be trusted and given more priority. When a node's TR value falls below the T_{evil}, it indicates a malicious node that should be excluded from the network. As seen in Figure 3.3, the additive component λ denotes the extent of increasing TR each turn and the subtractive component γ is responsible for showing the extent of decreasing TR each time.

In general, for classification problems there is a trade-off between detection rates and false positive rates. If there is a strong focus on preventing wrong judgements (Configuration 1 in Figure 3.3), the TR will be changed very slowly. The reaction

speed of the system is slow, but it can achieve a very low false positive rate. With a focus on fast detection and strong punishment (Configuration 2 in Figure 3.3) the system reacts fast and can yield a high detection rate however at the cost of increased false positives. λ and γ can be used to fine-tune the balance between desired detection efficiency and acceptable false positive rate (Configuration 3 in Figure 3.3). In general the following relations hold

$$\begin{aligned} \lambda \uparrow \gamma \uparrow \quad &\Rightarrow \quad \text{Reaction Speed} \uparrow \\ \lambda \uparrow \quad &\Rightarrow \quad \text{False Negative} \uparrow \\ \gamma \uparrow \quad &\Rightarrow \quad \text{False Positive} \uparrow \ \text{Detection Rate} \uparrow \end{aligned} \tag{3.2}$$

n_{down} denotes the number of times that the TR is decreased and n_{up} is the number of times that the TR is increased. A node will be excluded from the system if

$$T_{initial} + \lambda \cdot n_{up} - \gamma \cdot n_{down} \leq T_{evil} \tag{3.3}$$

Respectively a node will be considered as a friend by another node if

$$T_{initial} + \lambda \cdot n_{up} - \gamma \cdot n_{down} \geq T_{friend} \tag{3.4}$$

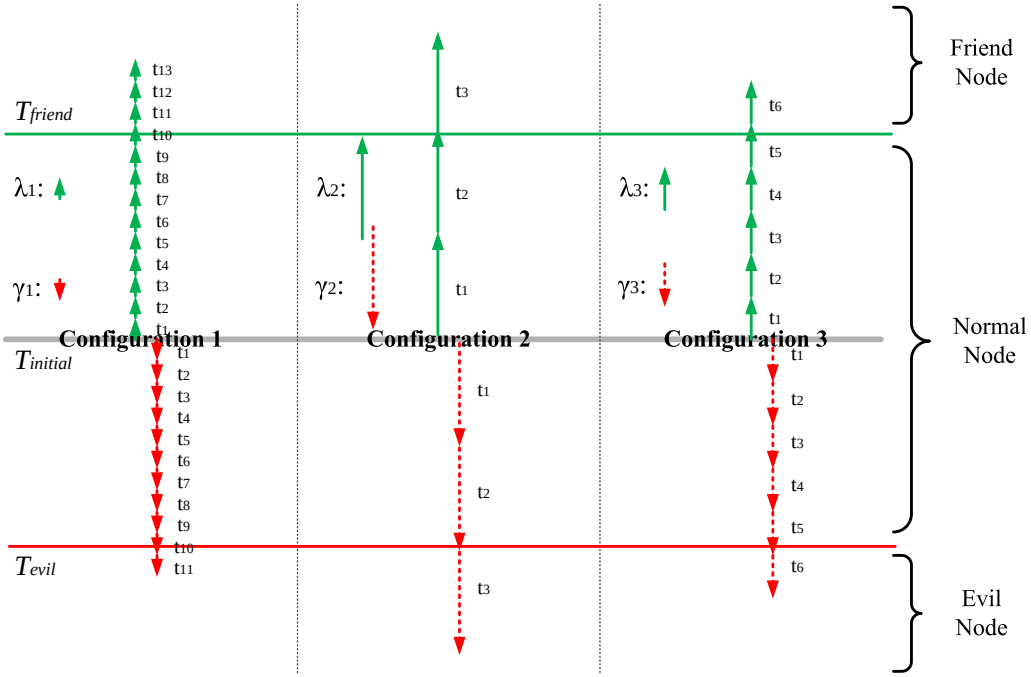


Figure 3.3: **Comparison of Different Strategies to Update TR**

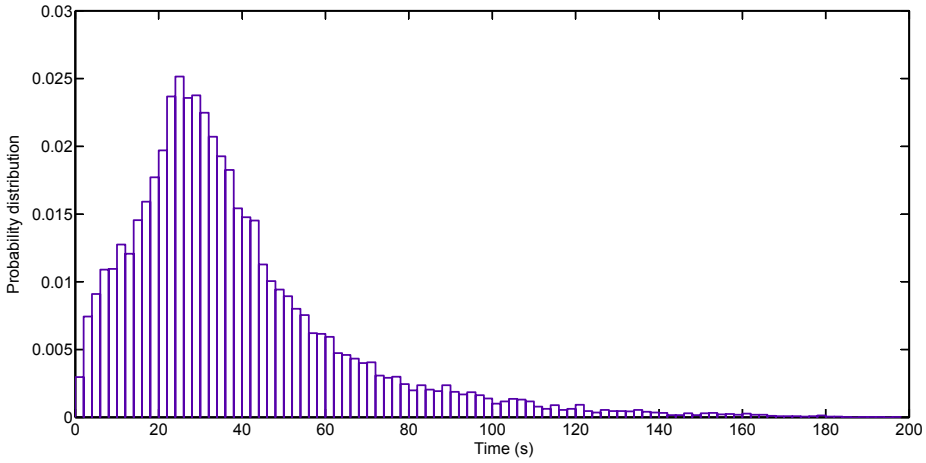
To prevent false positives and keep detection rates, the system makes sure to punish or reward nodes that clearly exhibit malicious or exemplary behavior by modifying their TR ratings. As we mentioned before, the $T_{initial}$, the T_{friend} and the T_{evil} are all static parameters in our system. Reasonable values for the parameters such as λ and γ will be chosen empirically to fine-tune the balance between detection rates and false positive rates.

3.3.2.3 Upper Bounding of the Re Set

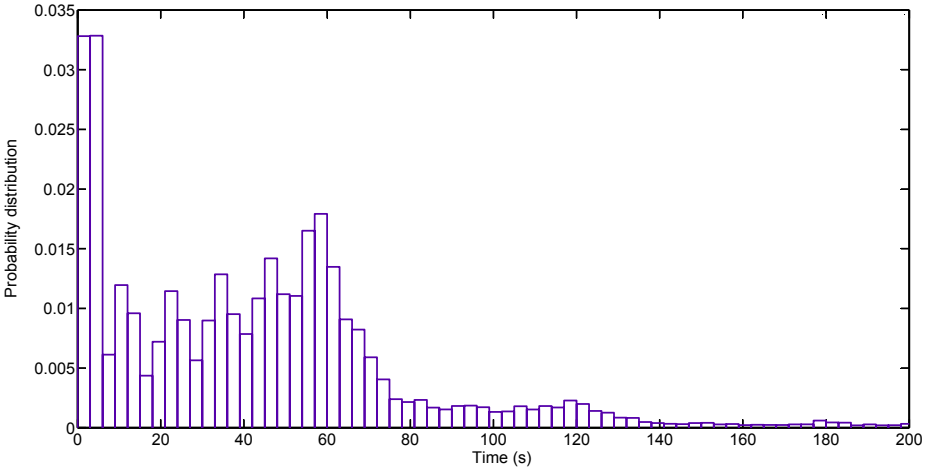
In the *Evaluation Module*, vehicular nodes update the TR of other encountered nodes based on the *Re* sets in the ERs provided by the encountered nodes. The *Re* sets record all the information about the exchanged messages. During every contact ERs are exchanged. The more messages get transferred between nodes the larger the *Re* set grows. Although in our MDS the *Re* set only consists of 2-tuples (*id*, *src*), occupying little space compared to other MDSs [15, 16, 17], it is necessary to place an upper bound on the size of the *Re* sets, as large *Re* sets can take up too much of a node's memory and generate more overhead when exchanging ERs between two nodes. Also colluding attackers might generate large *Re* sets, if they are colluding to fake a good reputation (see *Behavior 9* and *10* in section 3.2.2.2). This can cause severe problems when nodes judge the behavior of others. Hence, a limited upper bounding for the *Re* sets in ERs is imperative.

To obtain a reasonable value for the upper bounding of the *Re* sets, we first observe the connectivity time in three different scenarios. The connectivity time between vehicular nodes has a direct relation with the number of messages that can be exchanged during one contact. In this thesis, we use the relaxable definition for the connectivity time here: The connectivity time in this thesis denotes the maximum physical connectivity time. Usually when two nodes finish exchanging messages, they do not need to keep the physical connectivity anymore even though they are in each other's communication range. However, the connectivity time depicted in this section defines that, even though the vehicular nodes finish transmitting messages, they will keep the physical connectivity until the links totally fail. In other words, as long as nodes are in each other's communication range we define that they are connected. Using the idea of the "physical connectivity time", we can evaluate the maximum number of transmitted messages in each contact.

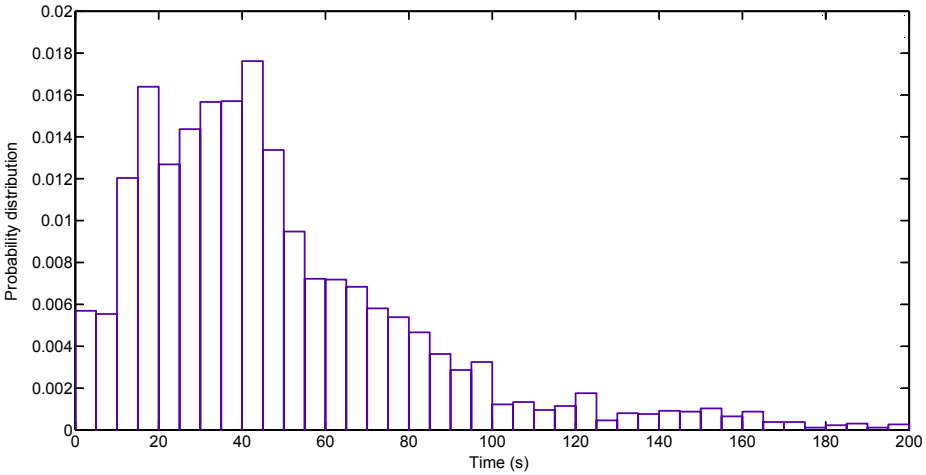
Figure 3.4 presents the distribution of connectivity time in three different scenarios. For detail information about the scenarios refers to section 5.3.1. The x-axis in Figure 3.4 shows the connectivity time and y-axis is used to show the distribution proportion of the different connectivity time. Using the idea of maximum physical connectivity time, the distribution of connectivity time relies only on the relative position between the vehicular nodes.



(a) The Helsinki scenario



(b) The San Francisco scenario



(c) The Braunschweig scenario

Figure 3.4: **The Distribution of Connectivity Time**

According to the results depicted in Figure 3.4, we find that the distribution of the connectivity time can be approximated to the Gamma distribution as formula (3.5):

$$\begin{aligned} Ga(x) &= \frac{1}{\beta^{\alpha}\Gamma(\alpha)} x^{\alpha-1} e^{-\frac{x}{\beta}} \quad x > 0 \\ \Gamma(x) &= \int_0^{\infty} t^{x-1} e^{-t} dt \end{aligned} \tag{3.5}$$

The Gamma distribution consists of two parameters: A shape parameter α and a scale parameter β. When a data set follows the Gamma distribution, the mean of the data set is equal to $E[X] = \alpha\beta$, and the variance of the data set is equal to $Var[X] = \alpha\beta^2$. According to the results depicted in Figure 3.4, we obtain the (mean, variance) tuples of the data set: $(E[X], Var[X])_{hel} = (37.4678, 702.235)$, $(E[X], Var[X])_{sf} = (42.3859, 1256.341)$ and $(E[X], Var[X])_{bs} = (47.4144, 1154.796)$. We captures the closest relation between the connectivity time interval and the probability distribution in Figure 3.5. The fitting parameter α and β used in Figure 3.5 are $\alpha_{hel,sf,bs} = \{2.0924, 1.04082, 1.89387\}$ and $\beta_{hel,sf,bs} = \{17.9066, 40.7236, 25.0357\}$. According to the formula (3.5), the (mean, variance) results depicted in Figure 3.5 are equal to $(E[X], Var[X])_{hel} = (37.4678, 670.922)$, $(E[X], Var[X])_{sf} = (42.3859, 1726.110)$ and $(E[X], Var[X])_{bs} = (47.4144, 1187.050)$.

The results in Figure 3.5 show that when the connectivity time interval is between 1 s and 100 s, the connectivity time has a overwhelming probability distributing in this interval. In other words, the major connectivity time is distributed below 100 s. When the connectivity time is longer than 100 s, the results show that there is a very low probability that the connectivity time between nodes is bigger than 100 s. In our thesis, the transmitting speed of vehicular nodes is 250 kB/s. When the connectivity time follows the major connectivity time distribution ($t \leq 100$ s), we choose the maximum connectivity value of 100 s for one contact, therefore, the maximum amount of data that will be transferred in the contact has the value of 25000 kB (250 kB/s $\times$ 100 s = 25000 kB). The message size in the system is between 500 kB and 1 MB. Hence, the maximum number of transferred messages is $N_{max} = 50$ (25000 kB $\div$ 500 kB = 50) in one encounter. The connectivity time, which is more than 100 s, has a less pronounced representative of nodes' behavior. Although the proportion is smaller, we still need to consider this condition.

Hence, taking into consideration the influence of various factors, we ultimately set the size of Re to 100 ($S_{Re} = 100$), which is almost two times of the maximum number $N_{max} = 50$ in the major connectivity time distribution. In each contact, if the number of transmitted messages is bigger than S_{Re}, the system should randomly choose only S_{Re} message information to fill in the ERs. In the ERs, both of the Re sets ($Re_{i \to j}$

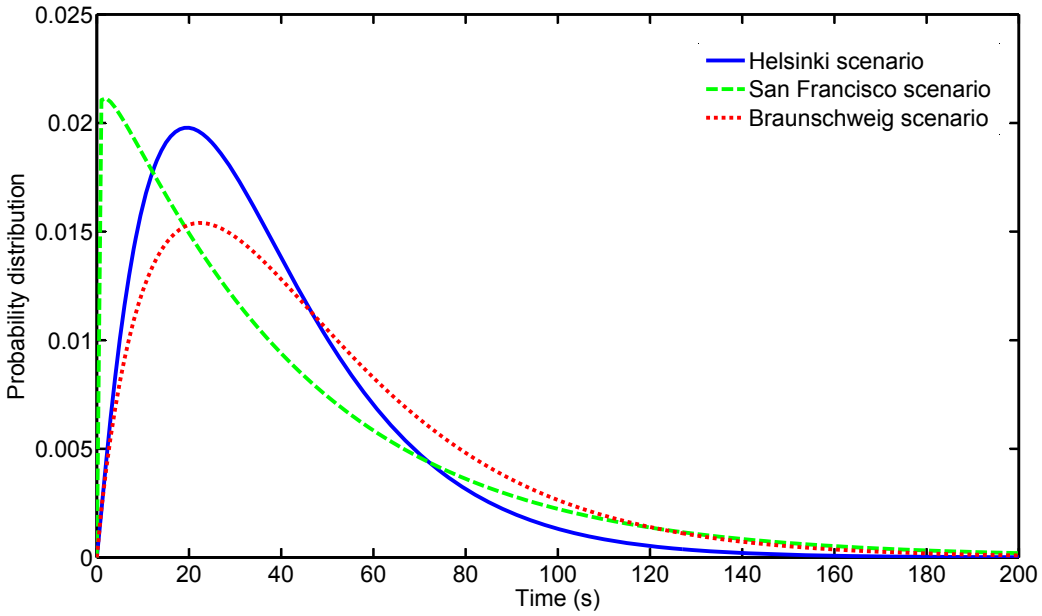


Figure 3.5: **Gamma Distribution**

and $Re_{j \to i}$) should follow the upper bounding principle. In the realistic scenario, the size of messages may be different according to different applications. However, we can apply the same approach to set the practical S_{Re} for the given application.

3.3.3 Decision Module

The *Decision Module* is responsible for making an appropriate decision after vehicular nodes receive an updated TR. Here vehicular node i and node j are used as an example to illustrate the work process of the *Decision Module*. Node i makes a decision for node j as follows: If the updated ${TR_i}^j$ is less than the T_{evil}, vehicular node i adds j to its LBL and refuses to exchange messages with j for a specified time. If the value of updated ${TR_i}^j$ is between the T_{evil} and the T_{friend}, vehicular node i will treat j as a normal node and transmit messages to j. If the updated ${TR_i}^j$ is more than the T_{friend}, it indicates vehicular node j forwards messages for others, therefore, vehicular node i will trust node j and list it as friend. When vehicular node i simultaneously communicates with multiple vehicular nodes, it will first transfer messages with its friend nodes such as node j, and then forward messages to normal nodes. Besides, vehicular node i will update the TR information of vehicular node j in its ML.

3.3.3.1 FBL Principle

When multiple nodes are in communication range, friend nodes will be preferred, i.e. they will be contacted first. We have looked upon improving the detection performance by including the blacklists from other nodes with a specially high TR into the decision process. We will extend the friend mechanism, enabling further cooperation between friend nodes in order to detect malicious nodes faster.

When nodes choose to apply the function of the FBL, the friend mechanism is active. When two vehicular nodes have a friend relation with each other, besides transferring messages, they also accept the other's LBL and add the entries to their own FBL. When a new entry is accepted and added to a node's FBL, it will not immediately lead to blocking of the accused node. Instead a node needs to receive the accused node x from the LBL of M distinct friend nodes, before it will refuse to exchange messages with vehicular node x. Thus, the FBL can increase detection rate without negatively affecting the false positive rate. In the evaluations presented in this thesis, M is chosen to be 3 ($M = 3$).

In the corner case, when two friend nodes exchange their LBL and find that one of them regards a vehicular node x as friend, while the node x is in the LBL of the other node, both nodes will keep a record about this disagreement, but not immediately change their opinion of the vehicular node x in question. Only after a node receives assessment of the node x contradicting his own view from N different friends, it will change its opinion about the node x. If a friend is accused N times by other friend nodes to be evil, it will be demoted to normal status. Similarly, if a node in the LBL is found to be the friend of at least N other friends, it will be removed from the LBL and reset to normal status. For the evaluations presented in this work we set $N = 3$. All nodes for which no information is received from friend nodes and whose TR is between the T_{evil} and the T_{friend} are considered normal and are acceptable for communication. In our later evaluations, we will show that using friend mechanism, friend nodes help each other to detect malicious nodes faster.

3.4 Performance Evaluations

In this section, we will first introduce the simulator which we use to evaluate the effectiveness of our MDS in VDTNs and then give a brief introduction about the evaluation metrics to evaluate the performance of our MDS.

3.4.1 The ONE Simulator

We use the DTN simulator the Opportunistic Network Environment simulator (*The ONE*, version 1.4.1) [105, 106] to evaluate our MDS. *The ONE* is a Java-based simulator specifically designed for DTNs. *The ONE* is a powerful tool for running DTN simulations with different routing protocols, generating mobility traces, visualizing simulations interactively in real-time and presenting the results after their completion [107].

The ONE supports many DTN routing protocols, such as Epidemic, MaxProp, PROPHET and Spray and Wait as described in section 2.2. As Spray and Wait is a limited-copy routing protocol, in *The ONE* we choose fixed number of replicas equal to 6 ($L = 6$) to balance delivery rates with the number of relayed messages. The nodes in the system will follow the rule of the DTN routing protocols to choose the available path for messages. *The ONE* has several movement models implemented for DTNs, for example the Helsinki scenario is a realistic Map Based Movement Model that can apply map data and constrain node to move along the streets and roads of the Helsinki city (see section 4.3.1). Additionally, *The ONE* offers an extensible simulation framework supporting the import mobility traces and events. In our later work, we import the San Francisco trace [108] and the Braunschweig trace [38] in *The ONE* to evaluate our MDS (see section 5.3.1). Hence, the nodes will move in the network depending on the movement models we choose in *The ONE*. According to the realistic requirements, *The ONE* can provide the service to generate messages with different sizes and different speeds. Additionally, when we use *The ONE* to implement our MDS, all the nodes store information about prior interactions with other nodes as defined in the *Vehicular Node Model* (section 3.2.1). Malicious nodes follow the *Attack Model* outlined in section 3.2.2.

3.4.2 Evaluation Metrics

We evaluate the performance of our MDS in terms of the following metrics. According to different requirements, different metrics are used in different conditions.

1) *Detection speed*: The detection speed is defined as how long does it take before each malicious node is detected at least *once* by a normal node. The detection speed is an important metric for the system. Only after a malicious node has been detected, measures to mitigate its impact can be taken. For example, with a fast detection speed, the system can reduce the amount of transmissions to attackers and therefore reduce the wasted energy.

2) *Detection rate*: We define the detection rate as the percentage of malicious nodes that have been detected by *all* good nodes. For a detection rate of 100%, we

require that *all* malicious nodes are detected by *all* normal nodes. The detection rate is defined as:

$$d\_rate = \frac{\#true\_positive}{\#normal\ nodes \times \#malicious\ nodes} \tag{3.6}$$

where a true positive is defined as the correct detection of one malicious node by one of the benign nodes.

3) *False positive rate*: The false positive rate is the percentage of good nodes that are mistakenly detected as malicious nodes, the false positive rate is defined as formula (3.7). The false positive rate denotes how clear the MDS can discriminate between good and malicious nodes.

$$fp\_rate = \frac{\#false\_positive}{\#false\_positive + \#true\_negative} \tag{3.7}$$

where a true negative is a benign node that has not been detected as malicious nodes by the system.

4) *Misclassification rate*: The misclassification rate denotes the percentage of wrong detections in relation to the total number of detections. In a real scenario the misclassification rate shows for all individual positive (malicious) classifications of all nodes, how often the detection is wrong.

$$mc\_rate = \frac{\#false\_positive}{\#false\_positive + \#true\_positive} \tag{3.8}$$

5) *Relayed messages*: The relayed message is the total number of relayed messages. The MDS should limit useless transmissions (to malicious nodes) and enable the system to invest its resources in form of energy and MAC layer capacity to support cooperating nodes.

6) *Delivery rate*: The delivery rate is the percentage of generated messages successfully delivered to their final destinations. The delivery rate is an indication of the service quality in VDTNs.

$$dr\_rate = \frac{\#delivered\ messages}{\#created\ messages} \tag{3.9}$$

7) *Latency time*: The latency time denotes the time delay experienced by a successfully delivered message.

8) *Overhead rate*: We define the overhead rate as how many hops a message needs to be successfully delivered to the destination.

3.5 Conclusions

We have presented a basic architecture of our MDS in this chapter. We describe the *Vehicular Node Model* and the *Attack Model* in detail. Additionally we introduce the fundamental components of the MDS – the *Rule Violation Checks*, the *Evaluation Module* and the *Decision Module*. At the end we discuss *The ONE* simulator used by our system and the metrics to evaluate our MDS. In the following chapters, we will use different strategies under different conditions to cope with malicious nodes.

Chapter 4

Adaptive Threshold-based Approach to Detect Attackers in Vehicular DTNs

4.1 Introduction

Detecting attackers is difficult in VDTNs, where connectivity is intermittent and long delays are actually the norm. Hence, we proposed a basic architecture of the MDS in Chapter 3. By collecting and securely exchanging data of previous encounters, a vehicular node can assess the trustworthiness of other vehicular nodes in order to detect malicious behavior.

The most important part in this chapter is that we propose a fixed threshold and an adaptive threshold mechanism taking into account the amount of information available to back a classification. Meanwhile, we introduce the corresponding update ratios supporting our threshold strategy and implement a TR update mechanism in the *Evaluation Module* of the MDS. Moreover, we apply the FBL mechanism, making nodes exchange classifications with their trusted nodes to boost detection efficiency.

The goal of our MDS is that we exclude malicious nodes from the network and prevent them from further disrupting the network. In the following, we will first describe the detail of our threshold-based MDS, and then we use the metrics such as detection rates, false positive rates and detection speed to evaluate the effectiveness of our MDS.

4.2 Evaluation Module

Our applied MDS is an encounter record-based reputation system. Encounter record-based means, after two nodes met and exchanged messages, a data structure (ER)

describing the encounter is created and mutually signed by both parties (see section 3.2.1.1). As the ERs provide some provable information about a node's behavior in the past, they will be stored in nodes' memory and exchanged upon each contact. In the *Rule Violation Checks* of our MDS (see section 3.3.1), a vehicular node will use the received ERs and its ML to check for any obvious violations of the constraints laid out in section 3.2.2.2.

In the *Evaluation Module*, the exchanged and collected ERs are used to assess and weigh the behavior of nodes against each other. Based on the results obtained from ERs the system changes a node's TR, which will ultimately lead to a node being excluded from the network if its TR gets too low. In this *Evaluation Module*, two indicators are calculated from ERs to characterize a node's behavior (see section 4.2.1). The threshold algorithm and subsequent rules to modify the TR are detailed further in the *TR Update* and the *Threshold Strategy*, where they have proven to adapt successfully to scenarios.

The *Decision Module* is responsible for making an appropriate decision after a node's TR gets updated. According to the updated TR, the encountered node will be defined as an evil, normal or friend node. We also utilize the friend mechanism, enabling further cooperation between friend nodes in order to detect malicious nodes faster.

Here we continue to use the vehicular node i and node j as an example to illustrate how the *Evaluation Module* works.

4.2.1 Update Ratios

Each node will calculate the *Message Forwarding Ratio* η and *Node's Own Message Forwarding Percentage* χ of encountered nodes and subject them to the threshold algorithm trying to separate normal behavior from malicious behavior.

Based on the *Re* sets in the ERs provided by node j, vehicular node i first figures out the *Message Forwarding Ratio* η of node j, which is the total number of messages that are sent out by j in w ERs over the total number of messages received by j. Node j provides w ERs: ER_0, ER_1, ..., ER_{w-1}. $N_{send}^{ER_0}$, $N_{send}^{ER_1}$, ..., $N_{send}^{ER_{w-1}}$ denote how many messages are sent out by j in ER 0, 1, ..., w-1. $N_{recv}^{ER_0}$, $N_{recv}^{ER_1}$, ..., $N_{recv}^{ER_{w-1}}$ denote how many messages are received by j in ER 0, 1, ..., w-1. The *Message Forwarding Ratio* η can be expressed as formula (4.1). A bigger η indicates a cooperative node, while a smaller η signifies the selfishness of a node.

$$\eta = \frac{\sum_{m=0}^{m<w} N_{send}^{ER_m}}{\sum_{m=0}^{m<w} N_{recv}^{ER_m}} \tag{4.1}$$

As blackhole and greyhole attackers drop messages, they tend to transmit more messages which are generated by themselves. Selfish nodes also prefer to transmit their own messages. To detect this, we use another mechanism to find malicious vehicular nodes. Again $N_{send}^{ER_0}$, $N_{send}^{ER_1}$, ..., $N_{send}^{ER_{w-1}}$ denote how many messages are sent out by j in ER 0, 1, ..., w-1. $N_{send}^{jER_0}$, $N_{send}^{jER_1}$, ..., $N_{send}^{jER_{w-1}}$ are the number of messages which j generated by itself and sent out in ER 0, 1, ..., w-1. The *Node's Own Message Forwarding Percentage* χ is defined as follows:

$$\chi \;=\; \frac{\sum_{m=0}^{m<w} N_{send}^{jER_m}}{\sum_{m=0}^{m<w} N_{send}^{ER_m}} \tag{4.2}$$

A bigger χ indicates that node j prefers to send its own messages.

4.2.2 TR Update

According to the update ratios of node j, its TR will be updated. The system will punish or reward nodes that clearly exhibit malicious or exemplary behavior by modifying their TR ratings. Algorithm 2 shows the outline of updating the TR. The detailed rules are as follows:

1) Vehicular node i first figures out the *Message Forwarding Ratio* η of node j as we introduced in section 4.2.1. If $\eta > N_{threshold}$, we can proceed to step 2. $N_{threshold}$ is determined dynamically depending on the amount of ERs available in section 4.2.3.2. If $\eta \leq N_{threshold}$ this indicates that vehicular node j may selectively drop messages. Then the TR of node j will be decreased according to formula (4.3).

$$TR_i^j \;=\; TR_i^j \;-\; \gamma \qquad (0 < \gamma < 1) \tag{4.3}$$

2) The second indicator taken into consideration is the *Node's Own Message Forwarding Percentage* χ. $\chi \geq NR_{threshold}$ indicates that node j prefers to send its own messages. To punish this kind of behavior, we use formula (4.3) to decrease the TR.

3) If step 1 and 2 simultaneously detect bad behavior of vehicular node j, instead of using formula (4.3) twice, we use formula (4.4) to decrease the TR.

$$TR_i^j \;=\; TR_i^j \;-\; \rho \qquad (2 \cdot \gamma < \rho < 1) \tag{4.4}$$

4) If neither of these steps detects abnormal behavior, this indicates that vehicular node j is not malicious and its behavior will be encouraged by increasing its TR

according to formula (4.5).

$$TR_i^j \;=\; TR_i^j \;+\; \lambda \qquad (\gamma < \lambda < 1) \tag{4.5}$$

Algorithm 2 requires the number of the eligible ERs is ≥ 10. If the number of the eligible ERs is < 10, our MDS will sustain the initial TR for the node and give the node much more time to accumulate its ERs, preventing mistakes from making decisions based on insufficient information.

To encourage more vehicular nodes to participate in the network, the additive component λ is larger than the subtractive component γ. Only when vehicular node i is convinced that node j behaves badly because both thresholds (step 1 and 2) are triggered, the larger subtractive component ρ is applied to decrease the TR of j. Reasonable values for these parameters can be chosen empirically. In the evaluations presented here we choose γ to be 0.04, ρ to be 0.09, λ to be 0.06 and $NR_{threshold}$ to be 0.7.

Algorithm 2: **Update TRs**

i will update TR_i^j for j based on ER_j received from Algorithm 1 (see *Rule Violation Checks* in section 3.3.1)

1: η = GetForwardingRatio(ER_j)
2: χ = GetOwnForwardingRatio(ER_j)
3: **if** $\eta > N_{threshold}(ER_j.length)$ **then**
4: **if** $\chi \geq NR_{threshold}$ **then**
5: $TR_i^j \;=\; TR_i{}^j \;-\; \gamma$
6: **else**
7: $TR_i^j \;=\; TR_i{}^j \;+\; \lambda$
8: **end if**
9: **else**
10: **if** $\chi \geq NR_{threshold}$ **then**
11: $TR_i^j \;=\; TR_i{}^j \;-\; \rho$
12: **else**
13: $TR_i^j \;=\; TR_i^j \;-\; \gamma$
14: **end if**
15: **end if**
16: TR_i^j = MAX(TR_i^j, 1.0)
17: TR_i^j = MIN(TR_i^j, 0.0)

4.2.3 Threshold Strategy

In this section, we propose a scheme associated with thresholds to detect attackers, where each vehicular node monitors the activities of other nodes relying on the ERs and detects an attack if there is deviation from expected behavior.

4.2.3.1 Fixed Threshold

In this part, our focus is on finding the optimum threshold to detect malicious nodes in VDTNs. We use the threshold to distinguish between malicious and normal behavior. In our system, an attacker cannot significantly disrupt the performance of the network while staying under the detection-thresholds. However, if an attacker exceeds the fixed detection-thresholds several times, it will be detected and classified as misbehaving node. A fixed threshold mechanism was used in our paper [109] as a simple way to assess other nodes' behavior.

Using the threshold mechanism, the key point is how the detection-thresholds should be chosen to balance the detection rate and the false positive rate of the system. To determine the optimum threshold under present conditions, we analyze the traffic of the whole network and obtain a basic knowledge of each node's relative behavior. According to the different characteristics of the normal and malicious nodes, we have adjusted detection-thresholds manually by running different simulations and testing the results (see section 4.3.1). As we introduce in section 4.2.1 (Update Ratios), we assume each vehicular node holds a maximum number of w ERs. In the fixed threshold mechanism, we define each vehicular node will at most save $w = 100$ ERs in its memory and we choose $N_{threshold} = 0.75$ and $NR_{threshold} = 0.7$ as the fixed detection-thresholds.

Using the fixed threshold mechanism, we can obtain a high detection accuracy, keep a fast reaction time and save the memory for vehicular nodes. However, one problem is that both of the thresholds in our MDS need to be set manually in order to get good detection results. Furthermore, when some nodes have more contact opportunities than others in the network, it will be a little difficult for the fixed threshold mechanism to find the optimal balance between detection rates and false positives. Hence, in the next section we propose an adaptive threshold mechanism to improve our MDS.

4.2.3.2 Adaptive Threshold

As we mentioned before, when some nodes have more contact opportunities than others, the fixed threshold mechanism cannot handle the situation well. To optimize this, depending on the number of ERs provided by the communication partner, a vehicular node chooses an adaptive threshold ($N_{threshold}$) for η which is applied in step 1 of Algorithm 2 by the *Evaluation Module* [110]. Upon connecting, two vehicular nodes first exchange their ERs prior to exchanging any application data. The ERs provided by a communication partner are used to assess its trustworthiness. The rationale is that a larger number of ERs contains more information and thus the threshold can be stricter because the risk to misjudge a node is smaller. However, in VDTNs the connectivity is intermittent, hence, the system cannot guarantee that every vehicular node has enough ERs. Therefore, in case a node cannot provide enough ERs, a more relaxed threshold is applied. This ensures that the system will not generate too many false positives while still maintaining the ability to detect obvious offenders.

Appropriate functions determining suitable $N_{threshold}$ values have been found using extensive simulations (see section 4.3.1). The results show that under the same drop probability, the normal nodes' *Message Forwarding Ratio* η increases as the vehicular nodes store more ERs in their memory. Meanwhile, the malicious nodes' *Message Forwarding Ratio* η decreases when the vehicular nodes provide more ERs under the same drop probability. We obtain different $N_{threshold}$ under the different drop probabilities. To achieve a high detection rate but also maintain a low false positive rate, we choose the minimal $N_{threshold}$ of different drop probabilities under the same number of ERs as the final $N_{threshold}$. The results indicate that the optimal $N_{threshold}$ can be approximated by formula (4.6):

$$N_{threshold} = b - \frac{a}{k} \quad \text{(a} > 0\text{, b} > 0\text{, } k\text{: the number of ERs)} \tag{4.6}$$

This formula captures the relation between the number of available ERs and the optimal $N_{threshold}$, where a and b are the fitting parameters. In Figure 4.1, $(a,b)_{Epidemic} = (3.6952, 0.8447)$, $(a,b)_{MaxProp} = (4.0025, 0.8807)$, $(a,b)_{PROPHET} = (4.0857, 0.8105)$ and $(a,b)_{Spray\ and\ Wait} = (3.0687, 0.8596)$. Figure 4.1 shows the resulting threshold functions for different routing protocols. As routing protocols differ in the way they forward messages, they lead to different "normal" values for the η ratio used in Algorithm 2 and thus different threshold functions are needed to guarantee optimal system performance under different routing protocols. As usually in

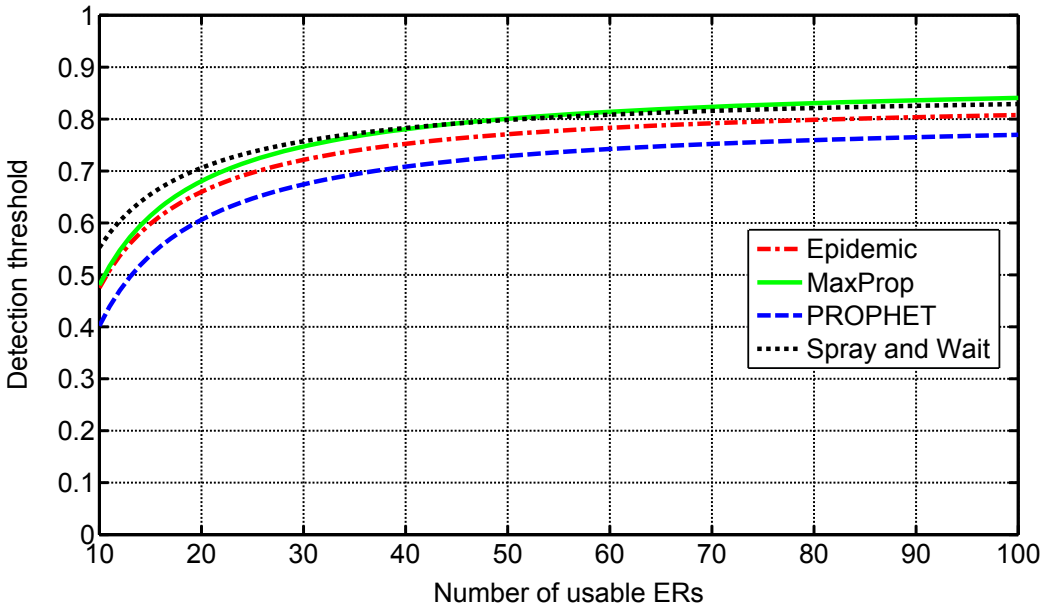


Figure 4.1: **Variable Detection Threshold**

a system the routing protocol can be seen as invariant, routing-dependent threshold functions are not a problem when deploying our system.

Nodes should store as many ERs as possible as evidence for their cooperating behavior. Figure 4.1 shows that if the number of ERs is above 80, the detection threshold is nearly stable. Considering the memory of vehicular nodes, we limit the maximum number of ERs (w) a nodes will store to 100.

4.3 Performance Evaluations

4.3.1 Simulation Setup

As a well-known and publicly available platform for DTN evaluations, *The ONE* DTN simulator (see section 3.4.1) is used to evaluate our MDS. In our simulations, 40 vehicular nodes with a transmission radius of 200 meters and a moving speed varying from 10 km/h to 50 km/h and a buffer size of 1 GB are uniformly deployed in the Helsinki city map within an area of 4500 m $\times$ 3400 m to simulate a VDTN. Each message has a lifetime of 45 minutes and a size between 500 kB and 1 MB. Vehicular nodes follow the shortest path map based movement. In an interval between 25 and 30 seconds, a source node which is randomly chosen will generate one message to a randomly chosen destination. We performed comparative measurements using the Epidemic, MaxProp, PROPHET and Spray and Wait routing protocols. We

randomly chose 4, 8 or 12 nodes (10%, 20%, 30%) among the 40 nodes as malicious nodes whose drop probability varies from 0.5 to 1. These malicious nodes follow the *Attack Model* outlined in section 3.2.2. They will independently attack the system and also have the ability to collude with each other as described before. The simulation time is 12 hours (43 200 s). Unless otherwise noted the results presented for each scenario are the average, min and max results of 10 experimental runs. Vehicular nodes apply the MDS starting from second 10000. The basic parameters for the Helsinki scenario are given in Table 4.1.

Table 4.1: **Simulation Parameters in the Helsinki Scenario**

Value	**Helsinki**
# nodes	40
Area	4500 m × 3400 m
Transmission radius	200 m
Movement	map-based shortest path
Buffer size	1 GB
Simulation time	12 h
Malicious nodes	10%, 20%, 30%
Drop probabilities	0.5-1.0

In our simulations, we use two strategies, the fixed threshold and the adaptive threshold, to evaluate our MDS under four different routing protocols. Firstly, we choose a fixed threshold to detect attackers, showing whether our MDS can make an accurate decision. Then we will use an adaptive threshold to evaluate our MDS when there are blackhole and greyhole attackers in the system.

4.3.2 Using Fixed Threshold to Cope with Attackers

In this section, we propose our MDS using the fixed threshold strategy. Each vehicular node can independently detect malicious and selfish vehicular nodes, isolate them and prevent them from disrupting the network. We will evaluate whether our MDS can achieve a high detection rate and a low false positive rate when detecting blackhole and greyhole attackers. Moreover we will consider the detection speed of our MDS.

4.3.2.1 Detection Rate

Figure 4.2 demonstrates the detection rates versus different number of malicious nodes using different drop probabilities under four different routing protocols. The

x-axis shows the drop probability of attackers. When the drop probability is $p = 1$, it denotes a blackhole attacker. Drop probabilities $0 < p < 1$ characterize a greyhole attacker. The y-axis shows the MDS's detection rate which is defined in section 3.4.2. To better understand the performance of our MDS the average and the maximum and minimum detection rates from 10 runs are shown in Figure 4.2.

The results depicted in Figure 4.2 show that, when greyhole attackers adopt a high drop probability ($p \geq 0.7$), our MDS can sustain an average detection rate more than 92.8% in all conditions for the unlimited-copy routing protocols – Epidemic, MaxProp and PROPHET. However, when there are 10% attackers with the drop probability of 0.5 in the system, our MDS achieves an average detection rate of 74.3%, 70.2% and 72.6% for Epidemic, MaxProp and PROPHET respectively. Coping with malicious nodes with a low drop probability, the system achieves a lower detection rate, which is not as high as the detection rate of malicious nodes with a higher drop probability. The reason is that when malicious nodes adopt a high drop probability, there is a distinctive difference between the behavior of normal vs. malicious nodes, hence, it is easier for the MDS to detect them.

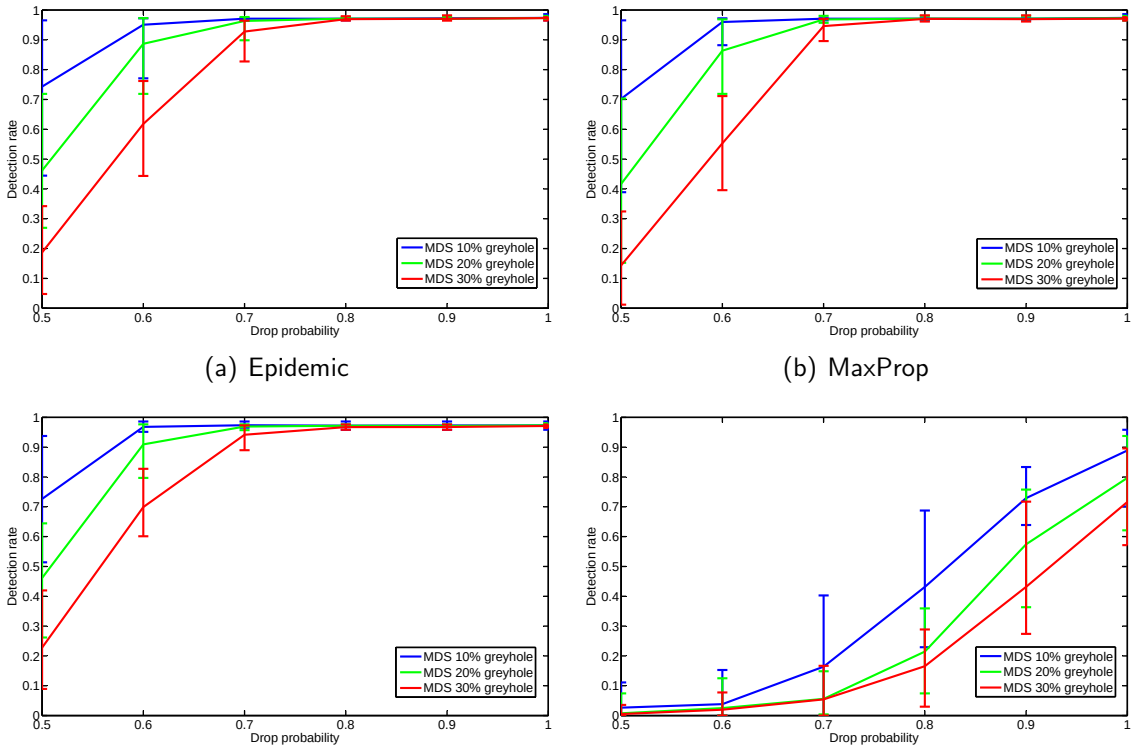


Figure 4.2: Detection Rate – Using Fixed Threshold in the Helsinki Scenario

However, for Spray and Wait in Figure 4.2 (d), with drop probabilities decreasing,

the detection rates significantly follow to decrease. When there are 10% blackhole attackers, the MDS sustains the average detection rate of 88.9%. As the same amount of greyhole attackers with the drop probability of 0.7 exits in the network, the average detection rate decreases to 16.4%. The reason is that compared to other routings, Spray and Wait relays fewer messages, obtains less information from ERs and thus cannot make a correct judgement.

Overall, it can be seen that the fixed threshold strategy is well suitable for the condition when the behavior between normal nodes and malicious node shows distinctive difference. When the drop probability is low, the behavior of malicious nodes is very similar to the behavior of normal nodes. This makes it harder for the fixed threshold strategy to discriminate between good and bad nodes.

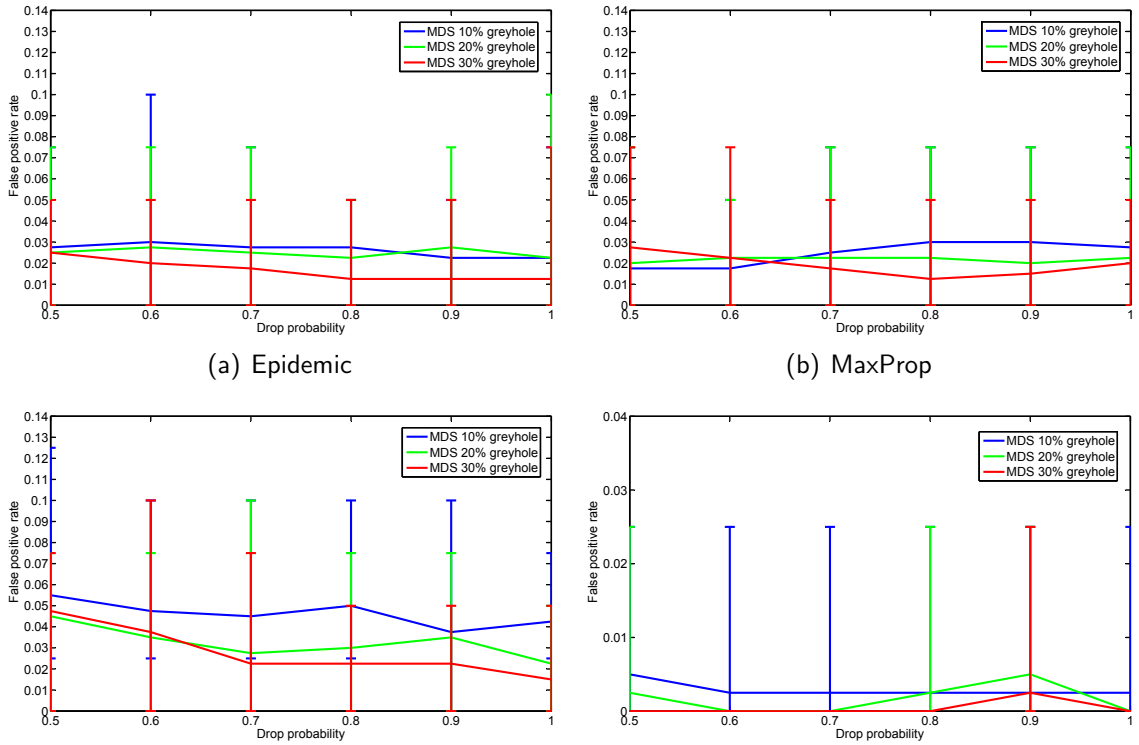
4.3.2.2 False Positive Rate

We use Figure 4.3 to show the false positive rates versus different percentage of malicious nodes under four different routing protocols. The x-axis shows the drop probability of attackers and the y-axis shows the false positive rate of the MDS (for the definition see section 3.4.2).

It can be seen, that our MDS can achieve a low false positive rate using different routing protocols in Figure 4.3: When the drop probability is high ($p \geq 0.7$), there is a distinctive difference between the behavior of normal vs. malicious nodes. Obtaining higher detection rates as shown in Figures 4.2 (a) to 4.2 (c), our system also achieves the average false positive rates below 2.75%, 3% and 5% under Epidemic, MaxProp and PROPHET respectively. As we mentioned before, when the malicious nodes adopt the low drop probability, the behavior of malicious and normal nodes is very similar to each other, therefore, it gets harder for the algorithm to discriminate between good and bad nodes and thus the false positive rate becomes a little larger. Considering the difficulty to detect attackers with a lower drop probability ($p < 0.7$), the MDS achieves higher average false positive rates compared to the average false positive rates of malicious nodes with a higher drop probability.

In Figure 4.3 (d), Spray and Wait sustains the lowest false positive rates compared to other routing protocols. However, this does not imply higher performance: Keep in mind, that using Spray and Wait the detection rate decreases significantly when attackers adopt a low drop probability in Figure 4.2 (d).

As shown in Figure 4.3, as the percentage of malicious nodes in the system increases, the average false positive rate will decrease. This is due to the fact that as the number of evil nodes increases, nodes have more opportunities to meet malicious nodes and thus can make a more precise decision.



(a) Epidemic (b) MaxProp

(c) PROPHET (d) Spray and Wait

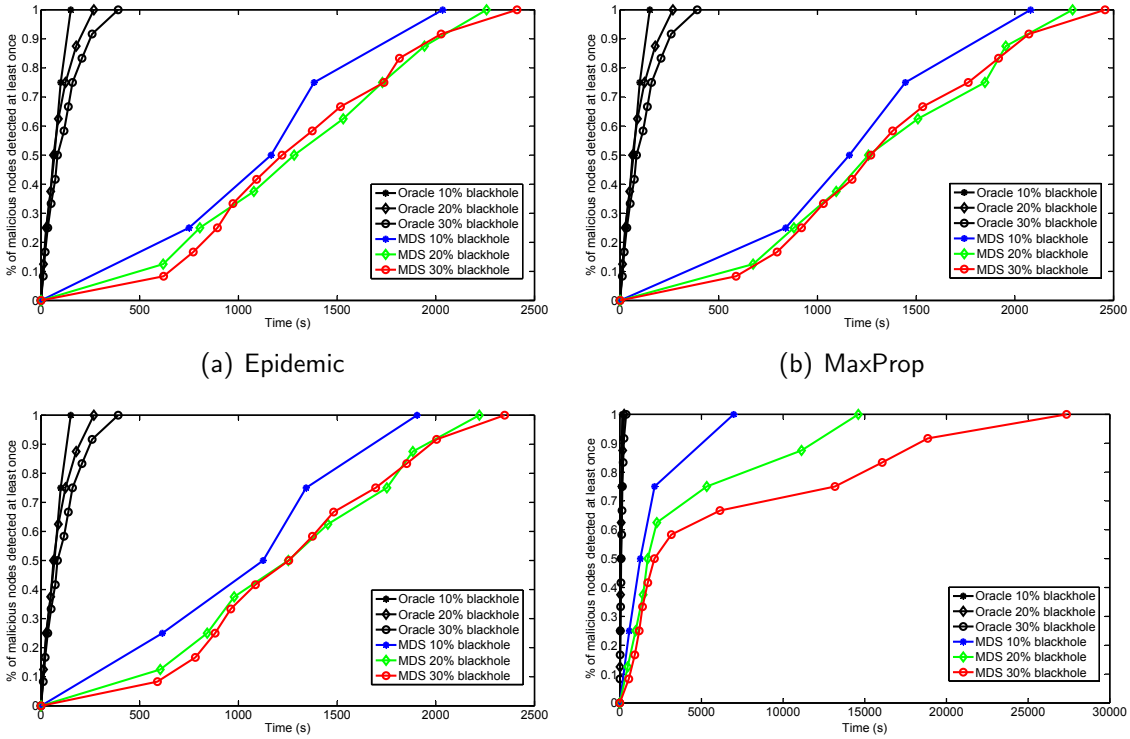
Figure 4.3: **False Positive Rate – Using Fixed Threshold in the Helsinki Scenario**

4.3.2.3 Detection Speed

The detection speed is an important metric for the system. Only after a malicious node has been detected, measures to mitigate its impact can be taken. Figure 4.4 shows the detection speed using four different routing protocols when only blackhole attackers exist in the system. A malicious node can only be detected after it met a normal node. Therefore, we use the "Oracle MDS" as a baseline for comparison: We assume that all evil nodes are known to the system beforehand. Hence, as soon as an evil node encounters a normal node it will be detected by the Oracle MDS.

From the detection speed results in Figure 4.4, it can be seen that the MDS takes about 2000 s to detect each of the malicious nodes using Epidemic, MaxProp or PROPHET when there are 10% blackhole attackers in the system. As the density of blackhole attackers increases from 10% to 30%, the MDS needs more time to detect all of the different malicious nodes. For routing protocols that exchange and generate less messages such as Spray and Wait, detection time is longer, because it takes a while until enough information has been accumulated in form of the ERs to let the

system detect the malicious nodes. In general, the MDS offers a good detection speed and tries to quickly exclude the malicious nodes from participating in the network.



(a) Epidemic (b) MaxProp

(c) PROPHET (d) Spray and Wait

Figure 4.4: Detection Speed – Using Fixed Threshold in the Helsinki Scenario

4.3.3 Using Adaptive Threshold to Cope with Attackers

In this section, we propose the adaptive threshold strategy to improve the performance of our MDS. We evaluate our method under the condition that the attackers can independently invade the network or collude with each other. We will test whether our MDS can efficiently detect evil nodes with varying drop probabilities, guaranteeing a high detection and low false positive rate while maintaining a low energy consumption. Furthermore, we will consider whether the presented approach can have a fast reaction time and maintain a high delivery rate.

4.3.3.1 Detection Rate

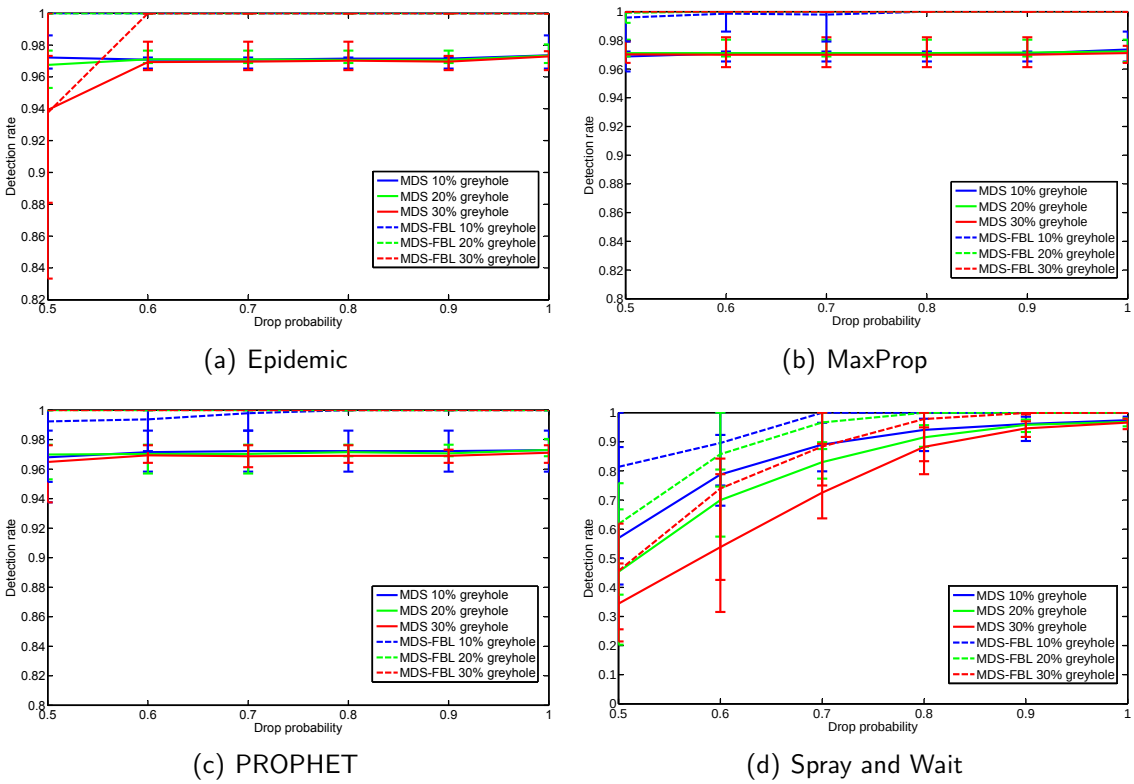
The results depicted in Figure 4.5 show the detection rates versus different number of malicious nodes with different drop probabilities using four different routing pro-

tocols. We plotted the average and the maximum and minimum detection rates from 10 runs. Our MDS is responsible for purging malicious nodes from the network as fast as possible. However, since malicious nodes with a high drop probability will have a more severe impact on the network performance, they should be detected quickly. As seen in Figure 4.5 (a) to 4.5 (c), coping with malicious nodes with a high drop probability ($p \geq 0.7$), our MDS performs well and can achieve a high average detection rate more than 97.0%, 97.0% and 96.9% under Epidemic, MaxProp and PROPHET. Compared to the detection rate of 92.8%, 94.6% and 94.2% using the fixed threshold strategy in Figure 4.2, the detection rates improve by 4.2%, 2.4% and 2.7%.

For the malicious nodes with a low drop probability that have a less pronounced effect on the network performance, our MDS can achieve an acceptable result but needs to take a longer time to detect all of them (see detection speed in section 4.3.3.4). When the attackers adopt a low drop probability ($p < 0.7$), our MDS obtains an average detection rate less than 97.2%, 97.1% and 97.1% under Epidemic, MaxProp and PROPHET. Under the current simulation time, the detection rate of malicious nodes with a lower drop probability is not as high as the detection rate of malicious nodes with a higher drop probability. However, compared to the MDS using the fixed threshold, the detection rates improve obviously when malicious nodes adopt a low drop probability.

The simulation results of Spray and Wait are shown in Figure 4.5 (d). It can be seen that when the drop probabilities decrease, the detection rates follow to decrease. When there are 10% attackers in the network, our MDS sustains a detection rate of 97.4% with blackhole attackers ($p = 1$) and a detection rate of 56.9% with greyhole attackers with the drop probability equal to 0.5 ($p = 0.5$). As we discussed before, Spray and Wait is a limited-copy routing protocol, relaying fewer messages and obtaining less information from ERs, hence, its detection rates are not as high as the detection rates of the unlimited-copy routing protocols under the same simulation time. However, the detection rates using the adaptive threshold are much better than the detection rates using the fixed threshold strategy.

In our simulation, we also evaluate the FBL principle (see section 3.3.3.1) in our MDS. When adding the FBL function, our MDS can achieve nearly 100% detection rates to cope with blackhole attackers under four different routing strategies. Varying the drop probability of greyhole attackers, our MDS with the FBL also achieves a higher detection rate for malicious nodes compared to the same condition applies without it.



(a) Epidemic (b) MaxProp

(c) PROPHET (d) Spray and Wait

Figure 4.5: Detection Rate – Using Adaptive Threshold in the Helsinki Scenario

4.3.3.2 Detection Rate with Colluding Attackers

We also studied how collusion between malicious nodes affects the detection rate. As described in the *Advanced Attacks* (section 3.2.2.2), colluding nodes will generate bogus, but correctly signed ERs. These attackers can choose to collude with other nodes in their communication range (*Behavior 9* in section 3.2.2.2) or they can join in a colluding group and apply the private key from its colluding group partners to generate bogus ERs at any time even though they are not in each other's communication range (*Behavior 10* in section 3.2.2.2). Figure 4.6 shows the detection rate in the scenarios with 10% to 30% blackhole attackers.

In Figure 4.6 (a), "MDS" denotes the condition when the blackhole nodes independently attack the system. "MDS-CO" denotes the condition when blackhole attackers collude to generate bogus ERs with other attackers in their communication ranges (*Behavior 9* in section 3.2.2.2).

When these blackhole attackers collude with each other, they forge beneficial ERs for themselves, which makes it more difficult to detect them. For the unlimited-copy routing protocols such as Epidemic, MaxProp and PROPHET, the tendency of the

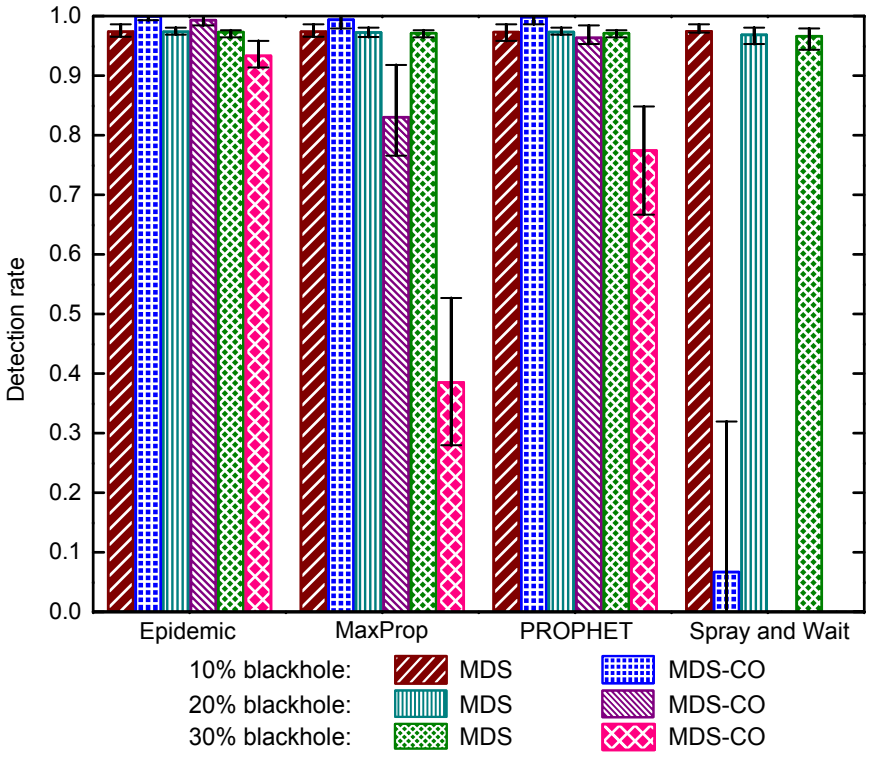
detection rates is not decreasing much. Using the unlimited-copy routing protocols, a lot of messages get transferred and a lot of ERs are generated. Therefore, when there are only few blackhole attackers colluding in the system, our MDS can still sustain the high detection rates. However, as the number of colluding attackers increases in the system, the detection rates also follow to decrease. The main reason is that in this case much more bogus ERs are created, because attackers are much more often beside each other. For instance, using the routing protocol PROPHET, the detection rates achieve 99.8%, 96.3% and 77.4% when there are 10%, 20% and 30% of the colluding blackhole attackers in the system respectively.

However, the detection rate goes down significantly using Spray and Wait. When the percentage of colluding attackers exceeds 20%, our MDS cannot detect these attackers. Compared to other routings, Spray and Wait relays fewer messages, obtains limited information from ERs. Hence, when the colluding attackers generate bogus ERs, these bogus ERs will affect the result of the update ratios and thus decrease the detection rates of the MDS.

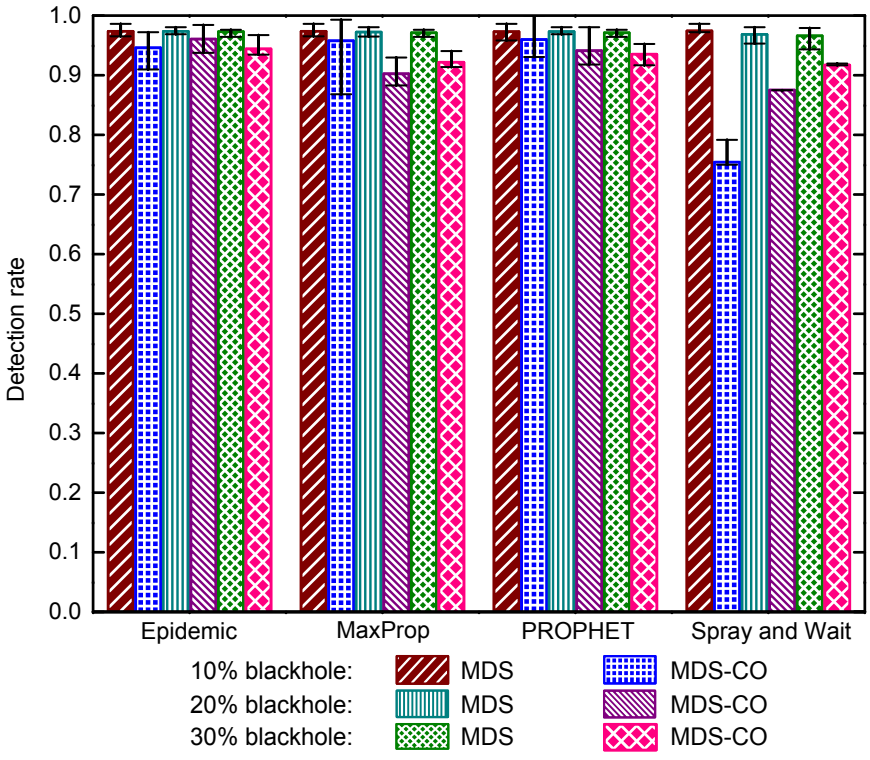
In Figure 4.6 (b), "MDS" also denotes the blackhole nodes independently attack the system. "MDS-CO" denotes the condition when blackhole attackers collude with each other and use *Behavior 9* and *Behavior 10* (see section 3.2.2.2) together to disrupt the network. Here, an attacker can not only generate bogus ERs with their encountered nodes, but also use the private keys from its colluding group to generate beneficial ERs for itself.

In Figure 4.6 (b), it is more likely that the MDS will use the *Rule Violation Checks* to detect rule violations due to inconsistent *sn* fields in the ERs and their MLs, hence, the detection rates increase compared to the results in Figure 4.6 (a). Potentially an advanced group of attackers can circumvent this, by synchronizing the *sn*s in the forged ERs with the *sn*s from real contacts, but this would require a real-time communication channel between attackers making such an attack much more complex.

Basically, the more frequently these advanced attackers interact with the system, the higher the chance of detection. Often attackers can obtain better ER results to cheat others, but there is still a non-negligible risk of being caught and punished, that provides enough incentive for most advanced attackers not to collude and forge beneficial ERs.



(a) Attackers using Behavior 9



(b) Attackers using Behavior 9 and Behavior 10

Figure 4.6: **Detection Rate for Blackholes with and without Collusion – Using Adaptive Threshold in the Helsinki Scenario**

4.3.3.3 False Positive Rate

Figure 4.7 presents the false positive rates of the system. It can be seen, that our MDS can achieve a low false positive rate versus different percentage of evil nodes using four different routing protocols. No matter which type of malicious nodes exists in the system, the malicious nodes with a high drop probability or with a low drop probability, our MDS tries to discriminate between good and bad nodes and keeps a low false positive rate.

As we discussed before, when the number of malicious nodes increases in the system, normal nodes have more opportunities to meet malicious nodes and thus can make a more precise decision. Hence, as the percentage of malicious nodes in the system increases, the average false positive rate decreases in Figure 4.7.

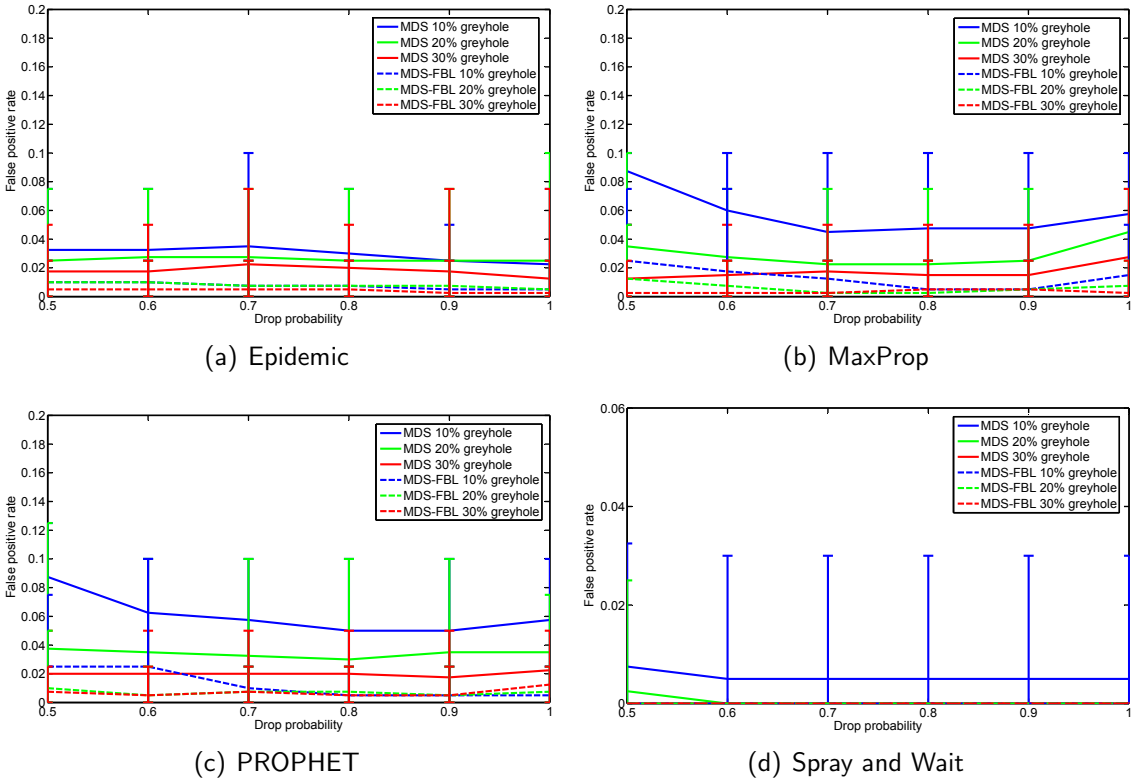


Figure 4.7: **False Positive Rate – Using Adaptive Threshold in the Helsinki Scenario**

When the MDS wants to achieve a higher detection rate, it will also face the problem with a higher false positive rate. It can be seen from the comparison between the detection rate results in Figure 4.5 and the false positive rates in Figure 4.7: When the drop probabilities are low, Epidemic, MaxProp and PROPHET keep high detection rates however obtain high false positives. For Spray and Wait, although its

detection rate is not so high, it can keep lower false positive rates compared to other routings.

In Figure 4.7, it shows that both MDS variants, with and without the FBL, can achieve a low false positive rate using different routing protocols. Using the FBL, the average false positive rate of our system is less than 2.5% in different scenarios. Firstly, the FBL does not increase the false positive rate, as the required consensus among 3 friends before labeling a node as malicious prevents the propagation of false positives through the system (see section 3.3.3.1). Meanwhile, comparing the MDS with the MDS with the FBL, the FBL can help friend nodes to make a final decision considering friends' suggestion, avoiding the false judgement about suspected nodes.

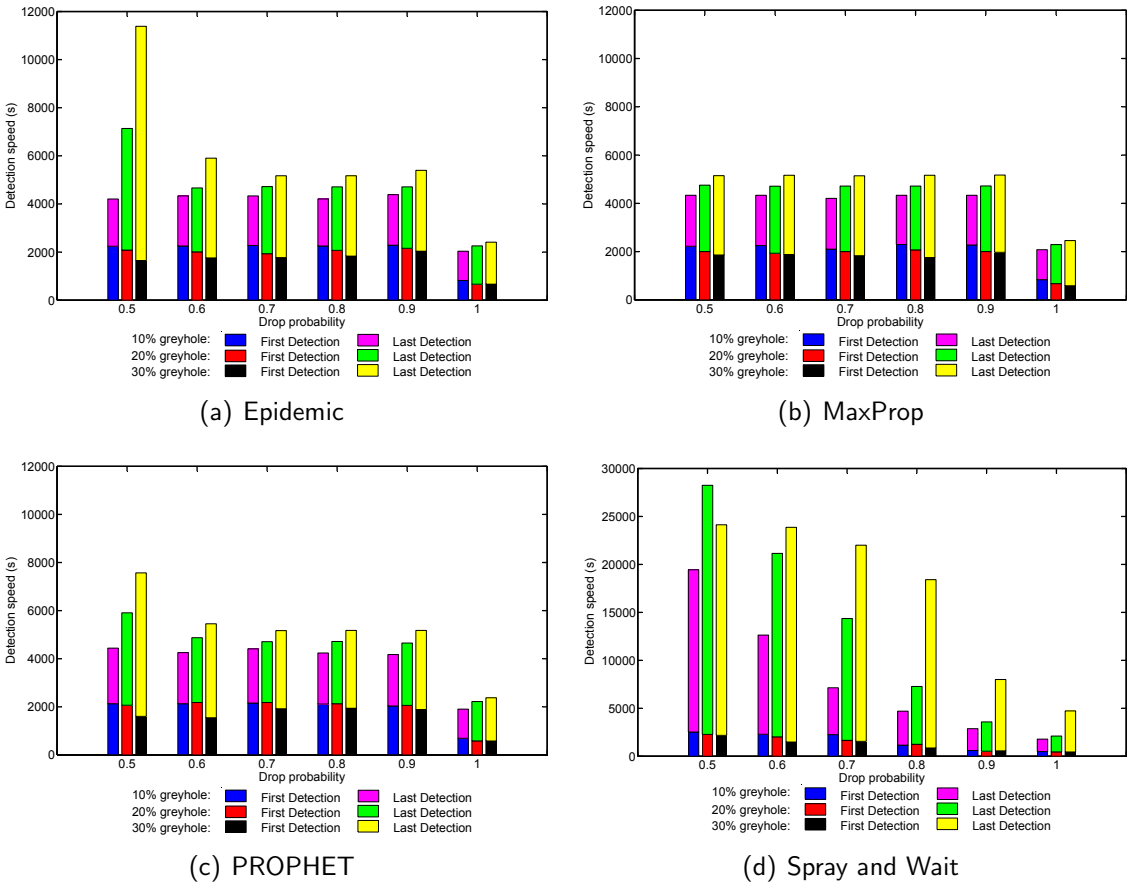
4.3.3.4 Detection Speed

Apart from a good detection rate, the detection speed is an important metric for the system. In Figure 4.8, we vary the drop probabilities as well as the routing mechanisms to check whether our MDS has a fast detection speed. The x-axis shows the drop probability used by the attackers and the y-axis shows the detection speed of the MDS (for the definition see section 3.4.2). "First Detection" is the time when the first malicious node is detected by a normal node. "Last Detection" is the time after which all of the malicious nodes have been detected at least once by a normal node. This time is also the lowest time boundary for achieving 100% detection rate, assuming the system could support perfect knowledge sharing among all nodes.

Figure 4.8 shows when 10% blackhole attackers ($p = 1$) exist in the system, the MDS takes about 2035 s, 2080 s, 1905 s and 1785 s to detect each of the malicious nodes using Epidemic, MaxProp, PROPHET and Spray and Wait respectively. Meanwhile, compared to the detection speed of greyhole attackers, the detection speed of blackhole attackers is always faster under the different routing protocols versus the different number of malicious nodes in the network because of their more obviously malicious behavior. When the attackers vary the drop probabilities from 1.0 to 0.5, the total detection speed takes longer. The reason is that facing the attackers with lower drop probabilities, the MDS needs to take a while until enough information has been accumulated in form of the ERs to let the system detect the malicious nodes.

Generally, detecting a higher number of attackers takes a longer time. Hence, as the density of the attackers increases from 10% to 30%, the MDS needs more time to detect all of the different malicious nodes.

For the limited-copy routing protocols that exchange and generate less messages such as Spray and Wait, the detection speed is longer compared to the detection speed



(a) Epidemic (b) MaxProp

(c) PROPHET (d) Spray and Wait

Figure 4.8: **Detection Speed – Using Adaptive Threshold in the Helsinki Scenario**

of the unlimited-copy routing protocols, because the MDS waits longer to accumulate enough information from the ERs, and then makes a judgement of the nodes.

For the detection speed, it makes no difference whether the system is exchanging LBL information with friends, as a node needs to be detected by at least one node before it has the chance to add to another node's FBL. Hence, we only show the detection speed without applying the function of the FBL.

In general the MDS offers a good detection speed and tries to quickly exclude the malicious nodes from participating in the network.

4.3.3.5 Relayed Messages

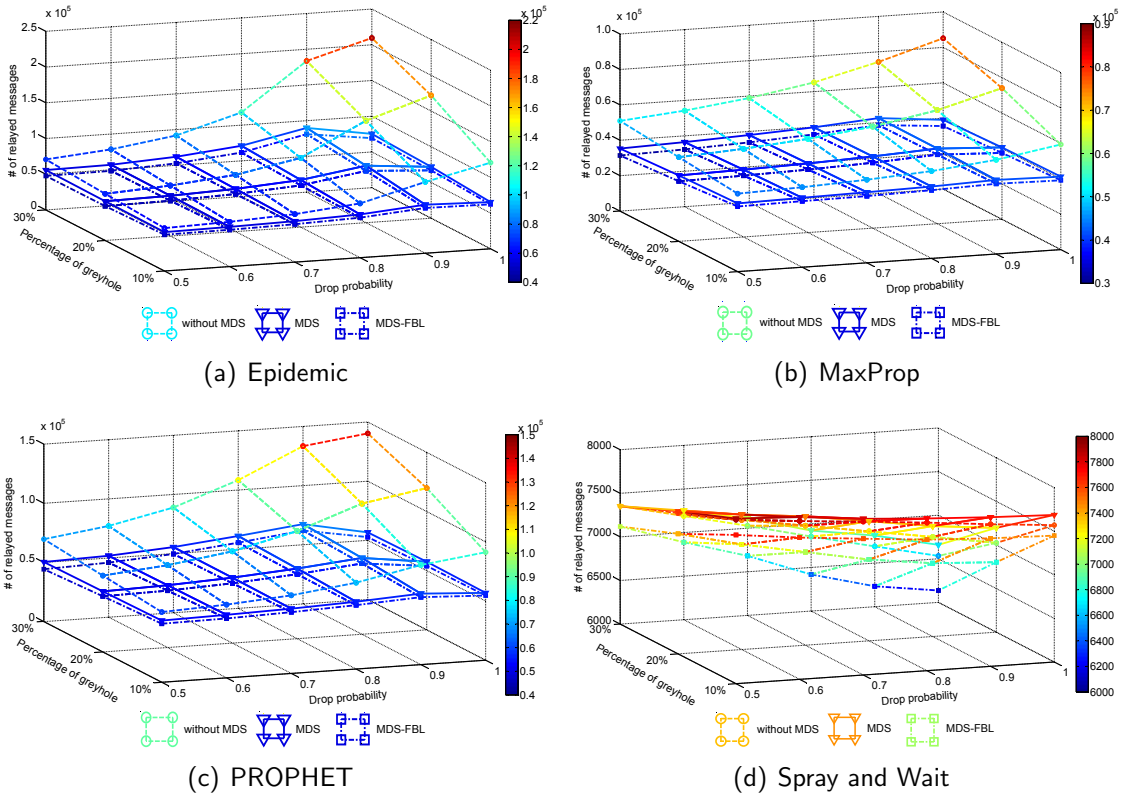
The energy consumption of a network is closely correlated to the number of transferred messages, as each message needs a certain amount of energy to be sent, received and processed. While energy usage might not be the primary concern in a VDTN,

less transferred messages also result in less congestion on the medium, and thus more capacity for legitimate messages.

Figure 4.9 presents the average number of messages relayed in the system versus the percentage of malicious nodes with different drop probabilities using four different routing protocols. The x-axis shows the drop probability; the y-axis shows the percentage of evil nodes in the system; the z-axis shows the total number of relayed messages in the system (for the definition see section 3.4.2). Because the number of relayed messages under four routing protocols shows significant difference with each other, hence, we choose different coordinate intervals for z-axis in Figure 4.9.

Figures 4.9 (a) to 4.9 (c) show that, especially for routing protocols with unlimited replication such as Epidemic, MaxProp or PROPHET, blackhole and greyhole attackers can cause a drastic increase in relayed messages. Epidemic spreads an unlimited number of message copies by transferring them to all other nodes they connect to. This simple approach floods the network with an unlimited number of message copies, leading to a high amount of energy wasted compared to other routings. When blackhole or greyhole nodes attack this kind of system, they will receive messages from other nodes during a contact, drop them afterwards and thus on the following contacts they may get many messages they once received again. We can see from Figure 4.9 (a), if there is no MDS in the system, a huge amount of relayed messages will be wasted by the attackers. In MaxProp, if the messages have been delivered to the destinations, MaxProp uses the acknowledgement to clear the messages in nodes' buffer and decreases useless transmission. Hence, the total number of transferred messages in MaxProp is decreased compared to Epidemic. In PROPHET, the nodes will transfer the messages to the nodes which are chosen according to its routing mechanism rather than transfer the messages to any random node. Hence, when blackhole or greyhole nodes attack these systems, they cannot affect the total number of transferred messages as drastic as it is the case for Epidemic. This can be seen in Figure 4.9 (b) and Figure 4.9 (c), the average number of transferred messages in MaxProp and PROPHET ($< 0.85 \times 10^5$, $< 1.4 \times 10^5$) is less than the average number of transferred messages in Epidemic ($< 2.09 \times 10^5$) when the MDS is not used.

In Figures 4.9 (a) to 4.9 (c), no matter varying the drop probability or changing the percentage of malicious nodes in the system, in these cases the MDS always maintains a lower number of relayed messages. When there are 30% of blackhole attackers in the system, the MDS achieves relayed message savings of 64.0%, 54.1% and 59.6% for Epidemic, MaxProp and PROPHET respectively. Adding the FBL mechanism leverages even more relayed message savings. Here 66.9%, 58.2% and 62.4% of relayed messages will be saved for Epidemic, MaxProp and PROPHET respectively. Hence, by detecting and excluding malicious nodes from the network, both MDSs save large amounts of energy and more importantly waste less capacity in the system.



(a) Epidemic (b) MaxProp

(c) PROPHET (d) Spray and Wait

Figure 4.9: **Relayed Messages – Using Adaptive Threshold in the Helsinki Scenario**

Routing protocols which limit the number of replicas such as Spray and Wait (see Figure 4.9 (d)) do not suffer so much from increased relaying. In the simulation setup each message is allowed to have 6 copies in our system, hence, the total number of relayed messages has an upper bound. When the malicious nodes drop the messages, the number of relayed messages will be decreased. If a MDS is effective in decreasing the chance that the limited replicas are relayed to the malicious nodes, the total number of relayed messages will be close to the upper bound. As can be seen in Figure 4.9 (d), no matter using our MDS or MDS with the FBL, our system can keep the total number of relayed messages close to the upper bound. For routing protocols with a limited number of replicas this is desired, as it increases the probability that messages can be forwarded towards their destinations and thus increases the delivery rate of the system.

4.3.3.6 Delivery Rate

The delivery rate is an indication of the service quality in VDTNs. For normal nodes the MDS should increase the delivery rate or at least not decrease it. The delivery

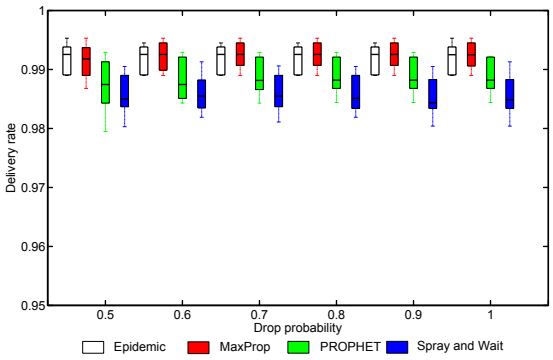
rate for malicious nodes should be decreased by excluding them from the network. Here we compared the delivery rate of normal nodes and malicious nodes to measure our MDS. We perform simulations under two different scenarios: One scenario with no resource constraints – Nodes have infinite buffer space, hence, nodes will not drop messages due to full buffers (Figures 4.10 and 4.11); another scenario with finite buffers (Figure 4.12).

We first evaluate our MDS and the MDS with the FBL when each node in the system has an unlimited buffer size. In our system, 1 GB is large enough for nodes, hence, it is used in the unlimited case. Figures 4.10 and 4.11 show the average delivery rate versus the number of malicious nodes in the system using different drop probabilities under four different routing protocols. The x-axis in Figures 4.10 and 4.11 shows the drop probability of attackers, and the y-axis shows the delivery rate of the nodes (for the definition see section 3.4.2).

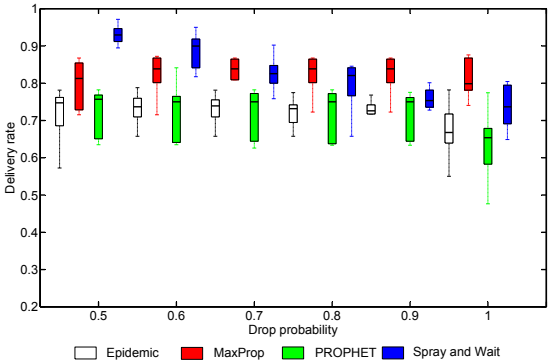
Here we compare the delivery rate of normal nodes and malicious nodes to measure our MDS and the MDS with the FBL. We observe that in Figures 4.10 (a), 4.10 (c) and 4.10 (e) and Figures 4.11 (a), 4.11 (c) and 4.11 (e), although the percentage of malicious nodes increases, the delivery rates of normal nodes are very high, achieving a delivery rate above 95% in all scenarios. These good results are due to the replication-based DTN routing protocols, the unlimited buffer space and the protection of our MDS.

In Figures 4.10 and 4.11, it can be seen that the MDS and the MDS with the FBL are able to keep the delivery rate of malicious nodes below the delivery rate of normal nodes. In Figure 4.10, although the system needs more time to independently detect blackhole and greyhole attackers, it does punish blackhole and greyhole attackers under the limited simulation time. Meanwhile, when friend nodes can exchange information with each other, we can see from Figures 4.11 (b), 4.11 (d) and 4.11 (f), that using Epidemic, MaxProp, PROPHET and Spray and Wait the average delivery rates of the blackhole and greyhole attackers sharply decrease. This shows that the blackhole and greyhole attackers are punished by our MDS or the MDS with the FBL. Therefore, to achieve a high delivery rate for their own messages, nodes need to transmit messages for other nodes.

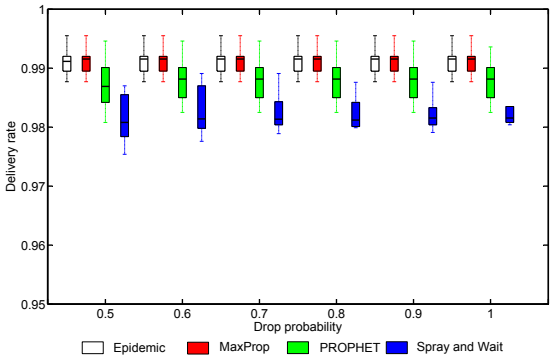
The results in Figures 4.10 and 4.11 show, that with the protection of our MDS or our MDS with the FBL, the attackers are not able to disrupt the operation of the network and non-cooperation does not pay off for selfish nodes, because they will get a lower service quality from the network for their own messages compared to cooperating nodes.



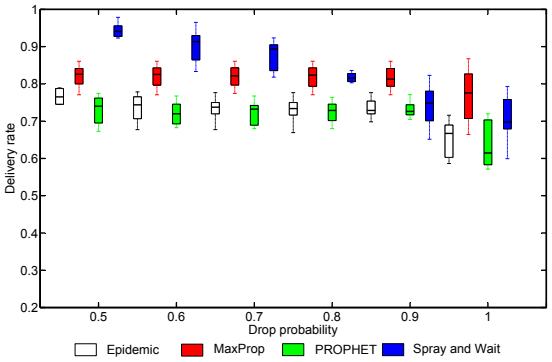
(a) Delivery rate of normal nodes with 10% attackers



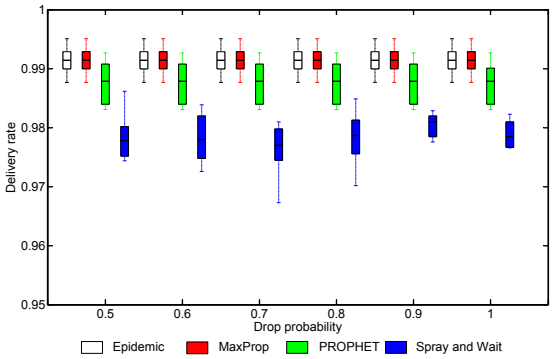
(b) Delivery rate of malicious nodes with 10% attackers



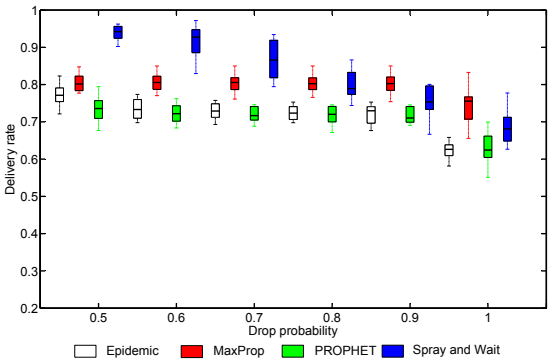
(c) Delivery rate of normal nodes with 20% attackers



(d) Delivery rate of malicious nodes with 20% attackers

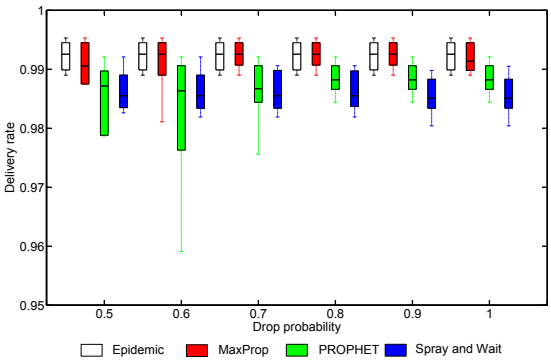


(e) Delivery rate of normal nodes with 30% attackers

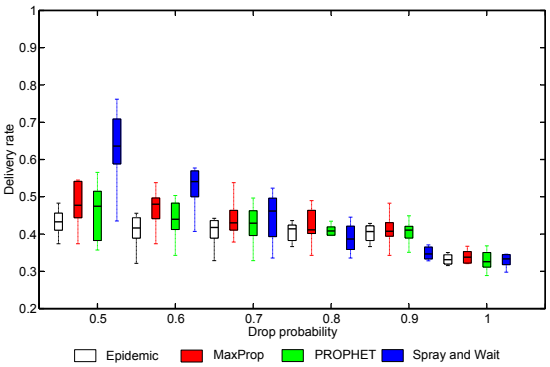


(f) Delivery rate of malicious nodes with 30% attackers

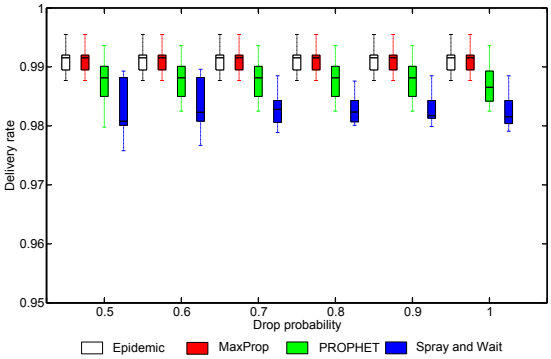
Figure 4.10: **Delivery Rate of the MDS Using Adaptive Threshold in the Helsinki Scenario**



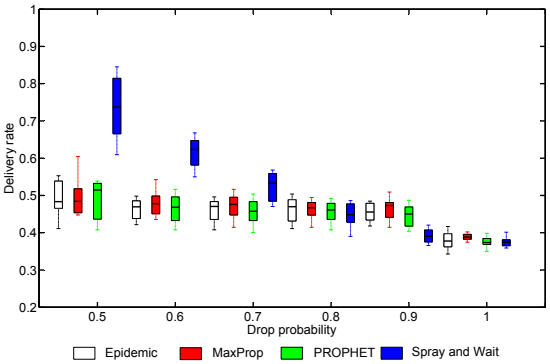
(a) Delivery rate of normal nodes with 10% attackers



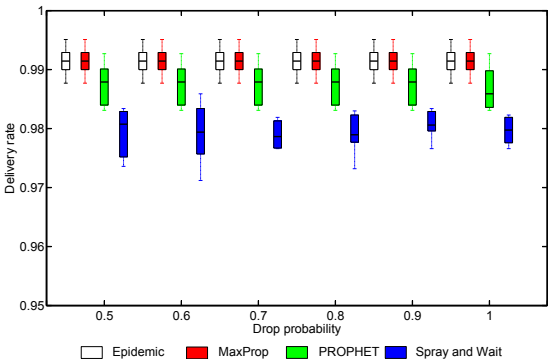
(b) Delivery rate of malicious nodes with 10% attackers



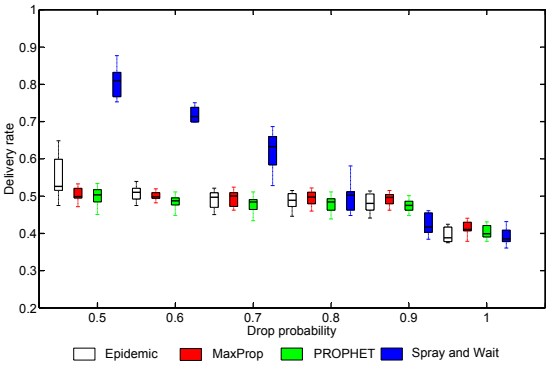
(c) Delivery rate of normal nodes with 20% attackers



(d) Delivery rate of malicious nodes with 20% attackers



(e) Delivery rate of normal nodes with 30% attackers



(f) Delivery rate of malicious nodes with 30% attackers

Figure 4.11: Delivery Rate of the MDS with the FBL Using Adaptive Threshold in the Helsinki Scenario

We also looked at the more realistic case of limited buffer sizes. The limited buffer size is especially a problem for flooding protocols such as Epidemic. Figure 4.12 shows the delivery rates of normal nodes and blackhole attackers using the plain MDS, the MDS with the FBL and without any MDS under Epidemic. In Figure 4.12, the x-axis shows the percentage of blackhole attackers in the system, the y-axis shows the delivery rate of the nodes. We vary the percentage of blackhole attackers and study the behavior of the nodes which have only 1 MB buffer size. It can be seen that without any MDS, blackhole attackers achieve the highest delivery rates. As the buffer of the nodes gets under pressure, messages will be dropped due to the limited buffer space. Since blackhole attackers only store their own messages, keeping forwarding their own messages will increase the probability of these messages to arrive to the destinations. Compared to blackhole attackers, normal nodes might need to transmit lots of messages from other nodes. Depending on the length of the contact, normal nodes might not even get the chance to transmit all of their own messages. This illustrates that in this scenario it is beneficial for the individual nodes to exhibit selfish behavior.

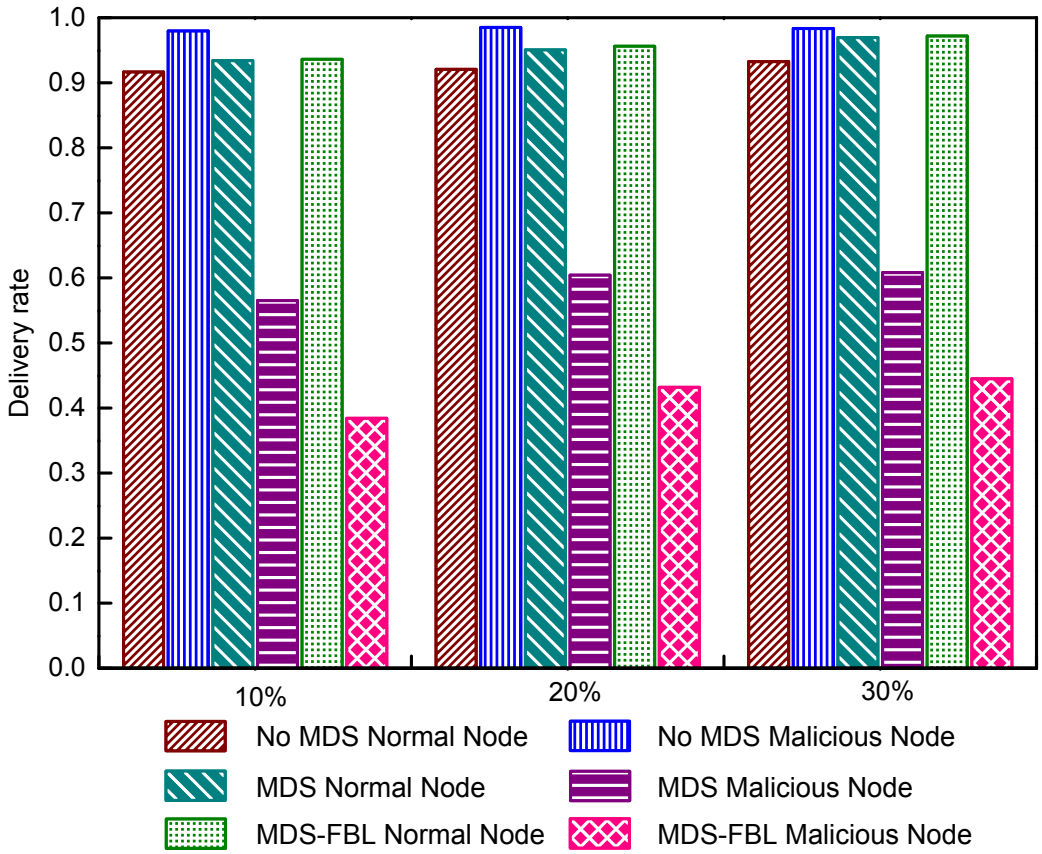


Figure 4.12: Delivery Rate of Epidemic with Constrained Buffer Size Using Adaptive Threshold in the Helsinki Scenario

Using the MDS and the MDS with the FBL the delivery rates for normal nodes go up, while the delivery rates of blackhole attackers go down. Furthermore, using the MDS and the MDS with the FBL, the delivery rate of normal nodes is higher than the delivery rate of blackhole attackers across all scenarios. Although before a blackhole attacker is detected it can still use the advantage of a non-pressured buffer as described above. However, once other normal nodes use the MDS to detect the blackhole attacker, it will be punished by other normal nodes and obtain low service from other nodes. Hence, it can be seen that when normal nodes apply the MDS, malicious nodes will achieve a low delivery rate provided by the other normal nodes. Furthermore, when the FBL extension is used, blackhole attackers will consistently achieve a much lower delivery rate. The FBL is shortening the detection time which limits the time during which a blackhole attacker is able to exploit the network.

Figure 4.12 illustrates that for constrained VDTNs, both our MDS variants, with and without the FBL, can tolerate the normal behavior of dropping packets when the buffers of the nodes are full and punish malicious behavior when the nodes drop the packets on purpose.

4.4 Conclusions

We presented a MDS that enables nodes in VDTNs to not only detect malicious nodes independently by distributing and combining information from previous encounters in the network, but also collaborate with their friend nodes in order to detect malicious vehicular nodes, even when some malicious nodes collude. The system excludes malicious nodes from the network, and thus prevents them from further disrupting the network. The integrated reputation system discourages selfish behavior. Applying a fixed threshold, our extensive simulations under different routing protocols show that our MDS can effectively achieve a high detection rate and a low false positive rate especially when the attackers demonstrate obviously malicious behavior. Using the adaptive threshold and the FBL mechanism our MDS can achieve a better network performance. The MDS is able to detect blackhole and greyhole attackers quickly and maintains a high delivery rate for normal nodes with unlimited as well as constrained buffer sizes. The presented approach can achieve a high detection rate and a low false positive rate for different scenarios where the number of malicious nodes, the attack intensity or the employed routing protocol is varied. Finally, our MDS can significantly reduce the number of wasting relayed messages to attackers for common DTN routing protocols, which create a large number of replicas.

However, it was still necessary to have an idea about a suitable value or function for the threshold before deployment, that could only be obtained by observing the system's normal behavior. This threshold was scenario- and routing-specific.

Chapter 5

Using Cluster Analysis to Detect Attackers in Vehicular DTNs

5.1 Introduction

In the previous Chapter 4, we implemented a MDS based on encounter records, in which nodes can detect and exclude blackhole and greyhole attackers from the network, without the need to have several nodes to vote on any decision. The adaptive threshold mechanism used by the MDS takes into account the amount of information available to back a classification. Meanwhile, utilizing the FBL mechanism nodes can exchange classifications with nodes they trust to boost detection efficiency. However, a suitable threshold could only be obtained by observing the system's normal behavior, making the threshold scenario- and routing-specific.

In this chapter we will introduce a cluster analysis approach that alleviates the need to determine thresholds before deployment and can work across a wider range of scenarios. We introduce a classifier based on clustering in [111, 112] to make the MDS work in different VDTN scenarios without a long training phase or prior knowledge about the application. Nodes will use cluster analysis to distinguish the behavior of encountered nodes and identify the malicious nodes. In this chapter we will first evaluate whether such a MDS can work efficiently in VDTNs when the nodes' behavior is homogeneous.

With the rapid development and proliferation of intelligent mobile devices such as smartphones, pedestrians carrying these devices might be used to act as networked nodes [113] enabling communication even when no other networking infrastructure is available. In this chapter, we will propose a hybrid urban communication network with buses, trams and pedestrians working together and evaluate the feasibility of creating such an integrated DTN consisting of smartphones and the public transportation system. More importantly, we will propose the MDS using cluster analysis

to protect the hybrid network and specially focus on the security issue of malicious and faulty nodes within this network [114].

5.2 Evaluation Module

In our MDS, we will first use the *Rule Violation Checks* (see section 3.3.1) for obtaining eligible ERs. These checks are based on the ML and the ERs provided by the encountering nodes and can detect the tampered ERs such as dropping or forging, which revealed through non-sequential sequence numbers or contradictions between the presented sn and the information in a node's ML about the last valid (sn, t) combination. For details refer to section 3.3.1.

The *Evaluation Module* will use these eligible ERs to determine the trustworthiness of other nodes. In the *Evaluation Module*, we will first discuss the *Update Ratios* which are the metric to represent the behavior of nodes. Afterwards, according to the different behavior of nodes, we introduce the basic notion about cluster analysis using K-means clustering to distinguish normal nodes from malicious nodes. Meanwhile, we also elaborate the improved TR update principle in this chapter.

The updated TR is delivered to the *Decision Module* which is responsible for making an appropriate decision after a node's TR gets updated. Once a node defines an encountering node as evil, it will refuse to exchange messages with this encountering node. When a node defines an encountering node as normal or friend, it will transfer messages with this encountering node. As friend nodes, the system allows nodes to apply the friend mechanism to boost the detection accuracy.

In this chapter, we still use vehicular node i and node j as an example to illustrate how the *Evaluation Module* works.

5.2.1 Update Ratios

In this part, we introduce the *Message Forwarding Ratio* θ and the *Message Receiving Ratio* ψ to describe the historical behavior of other nodes. These two indicators are calculated from the *Re* sets in the ERs.

According to the *Re* sets in the ERs from node j, node i first calculates the *Message Forwarding Ratio* θ of node j as formula (5.1). θ is the ratio between *forwarded* messages over the total number of *received* messages not destined for node j. $N^{ER_m}_{forwarded}$ indicates the number of messages forwarded to other nodes, but not originated from node j in the encounter record ER_m. $N^{ER_m}_{received}$ indicates the number of messages received but not destined for node j in the encounter record ER_m. A

bigger θ indicates nodes prefer transmitting messages for others, while a smaller θ signifies the selfishness of nodes.

$$\theta = \frac{\sum_{m=0}^{m<w} N_{forwarded}^{ER_m}}{\sum_{m=0}^{m<w} N_{received}^{ER_m}} \tag{5.1}$$

The second measure taken into consideration is the *Message Receiving Ratio* ψ. The idea is that nodes which drop packets will receive certain messages over and over again from different nodes. Therefore, the *Message Receiving Ratio* ψ is the ratio between received messages and the number of unique message *ID*s received. $N_{received}^{ER_m}$ is defined similarly to the definition of θ, while $received_{unique}$ is the number of *unique* messages received by this node:

$$\begin{aligned} received_{unique} &= |\{msg_{id} \mid \forall\, msg_{id} \in ER.Re_{x \to j}\}| \\ &\quad (x\text{: } j\text{'s encountered vehicular nodes}) \end{aligned} \tag{5.2}$$

The *Message Receiving Ratio* ψ is defined as formula (5.3). The rationale is that when a node receives msg_{id} n times because it keeps dropping that message, its $N_{received}^{ER}$ will increase n times while its $receive_{unique}$ will increase only by 1. For a perfectly behaving node without buffer pressure the best achievable value is $\psi_{opt} = 1$. Malicious nodes dropping messages will have higher ψ ratios.

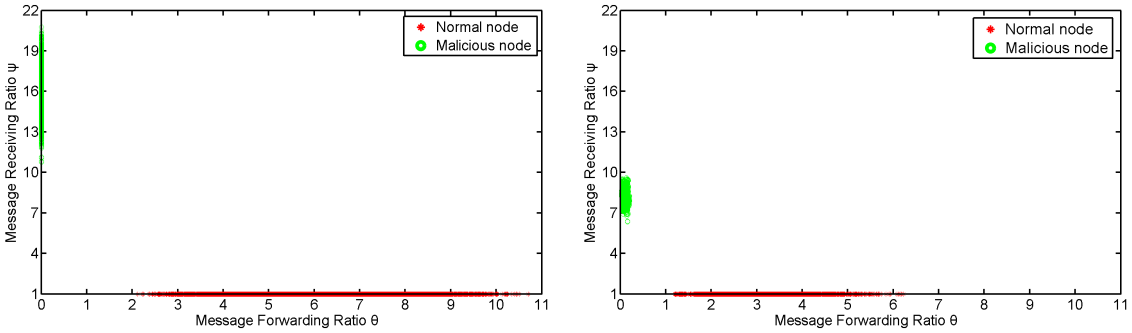
$$\psi = \frac{\sum_{m=0}^{m<w} N_{received}^{ER_m}}{received_{unique}} \tag{5.3}$$

5.2.1.1 Applicability of the θ and ψ Parameters for Misbehavior Detection

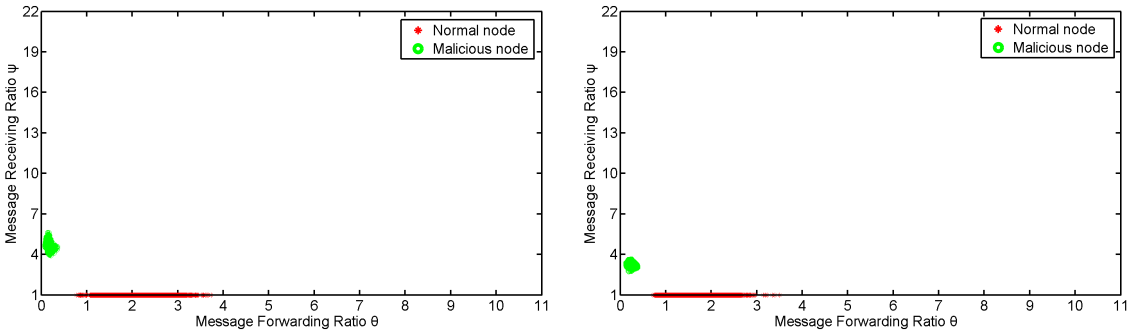
In Figures 5.1 to 5.4, we vary the drop probabilities as well as the routing mechanisms to get an idea about the expected (θ, ψ) tuples. We obtain the data by using extensive Helsinki scenario simulations where there are 20% evil nodes in the system and nodes run without our MDS (see section 5.3.1.1). The x-axis shows the nodes' *Message Forwarding Ratio* θ. The y-axis shows the *Message Receiving Ratio* ψ. The results in the figures show the (θ, ψ) values of the good and misbehaving nodes in all conditions.

The results depicted in Figures 5.1 to 5.4 show that, the (θ, ψ) tuple is an effective metric to represent the behavior of nodes. There is an obvious difference in the characteristic of the (θ, ψ) tuples between good and misbehaving nodes. When the drop probability varies from 1.0 to 0.5, the smaller the drop probability is, the similarity between the malicious node group and the normal node group becomes much closer to each other. The results demonstrate that when the attackers adopt

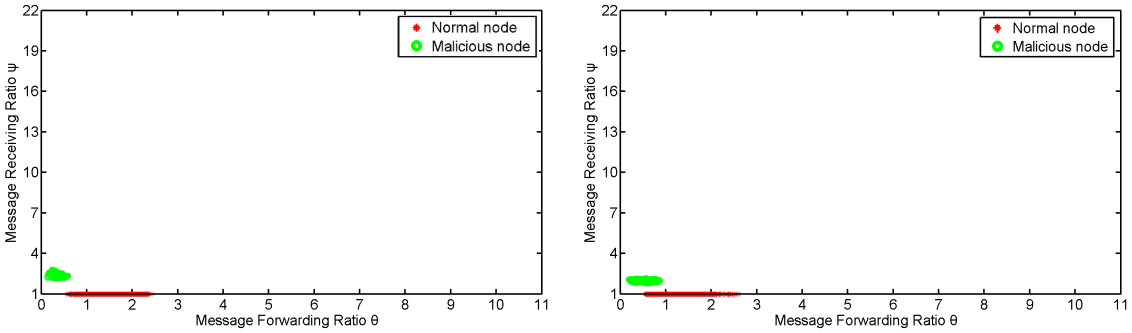
the smaller drop probability, they have similar behavior as normal nodes, increasing the difficulty for the MDS to detect malicious nodes. However, the results also show that the malicious nodes with lower drop probabilities will have a less pronounced effect on the network performance.



(a) Malicious nodes with the drop probability of 1.0 (b) Malicious nodes with the drop probability of 0.9

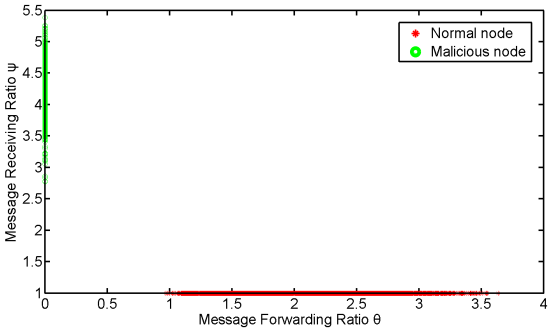


(c) Malicious nodes with the drop probability of 0.8 (d) Malicious nodes with the drop probability of 0.7

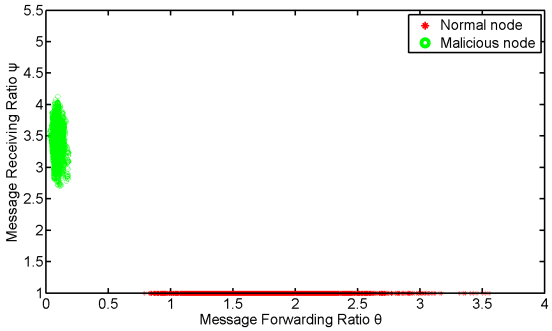


(e) Malicious nodes with the drop probability of 0.6 (f) Malicious nodes with the drop probability of 0.5

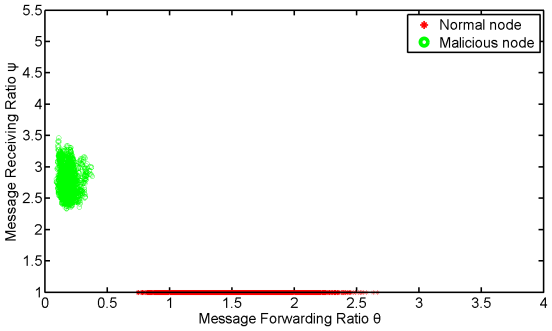
Figure 5.1: (θ, ψ) **Values Using Epidemic in the Helsinki Scenario**



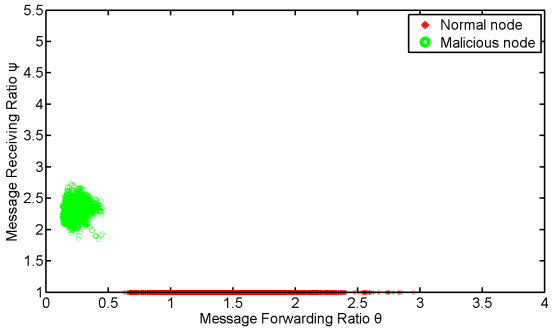
(a) Malicious nodes with the drop probability of 1.0



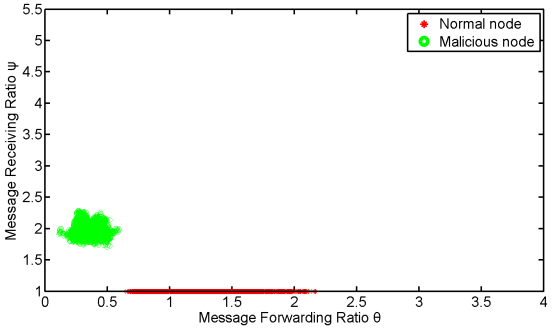
(b) Malicious nodes with the drop probability of 0.9



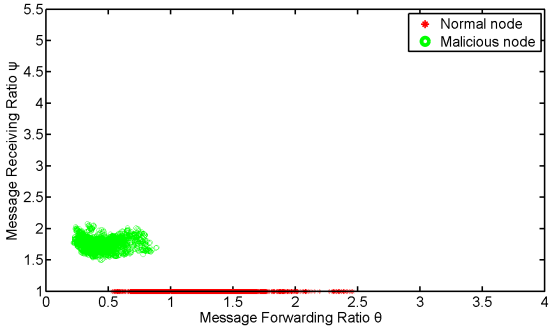
(c) Malicious nodes with the drop probability of 0.8



(d) Malicious nodes with the drop probability of 0.7

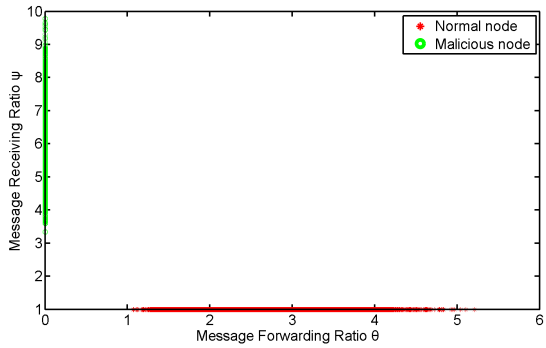


(e) Malicious nodes with the drop probability of 0.6

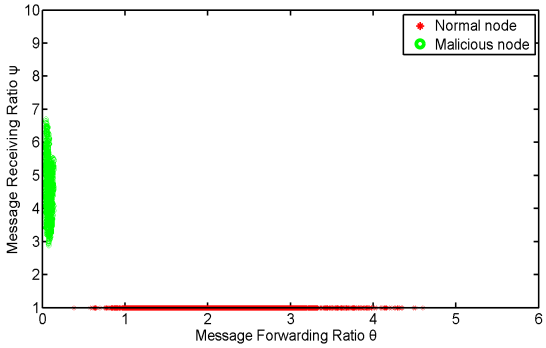


(f) Malicious nodes with the drop probability of 0.5

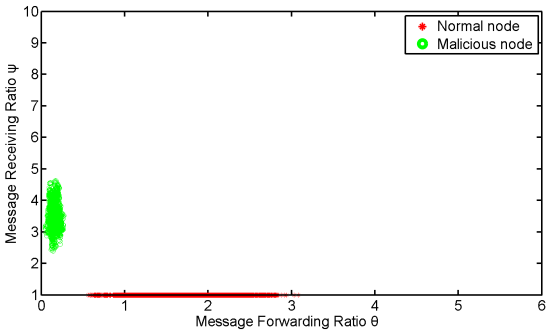
Figure 5.2: (θ, ψ) **Values Using MaxProp in the Helsinki Scenario**



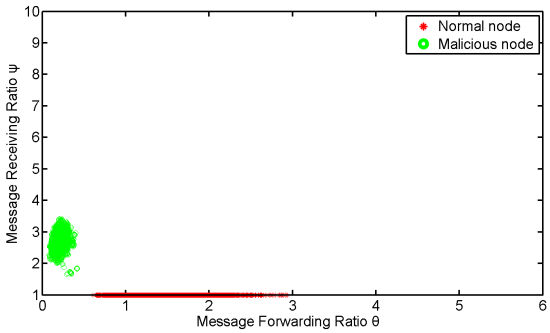
(a) Malicious nodes with the drop probability of 1.0



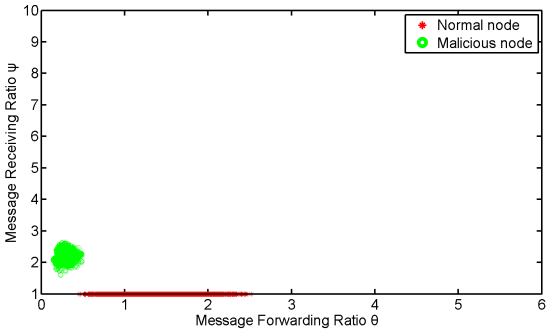
(b) Malicious nodes with the drop probability of 0.9



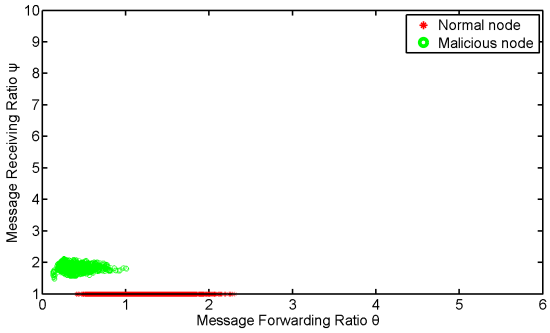
(c) Malicious nodes with the drop probability of 0.8



(d) Malicious nodes with the drop probability of 0.7

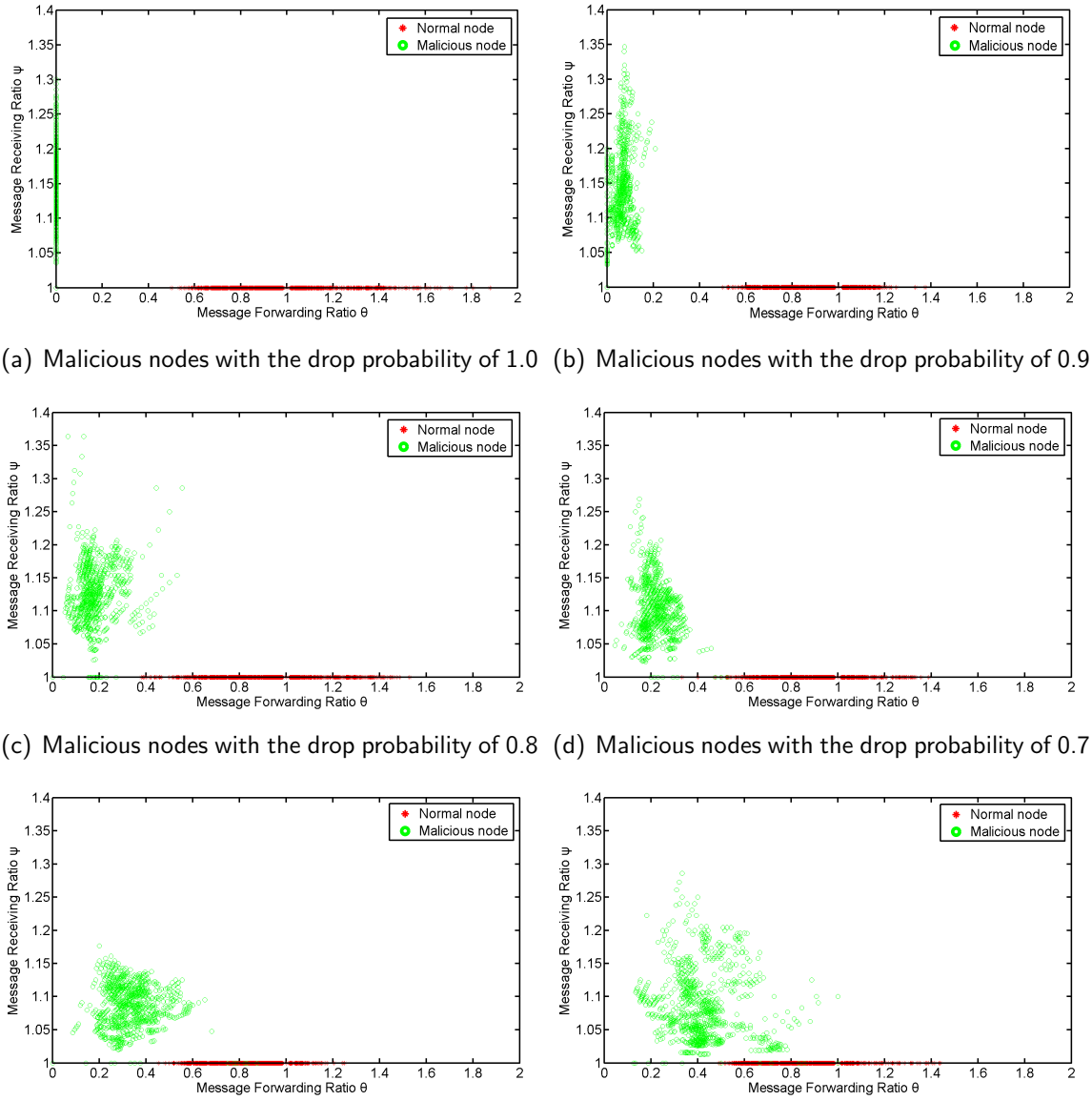


(e) Malicious nodes with the drop probability of 0.6



(f) Malicious nodes with the drop probability of 0.5

Figure 5.3: (θ, ψ) **Values Using PROPHET in the Helsinki Scenario**



(a) Malicious nodes with the drop probability of 1.0 (b) Malicious nodes with the drop probability of 0.9

(c) Malicious nodes with the drop probability of 0.8 (d) Malicious nodes with the drop probability of 0.7

(e) Malicious nodes with the drop probability of 0.6 (f) Malicious nodes with the drop probability of 0.5

Figure 5.4: (θ, ψ) **Values Using Spray and Wait in the Helsinki Scenario**

As seen from Figure 5.1 to Figure 5.3, for the unlimited-copy routing protocols we find that when there are blackhole or greyhole attackers in the network, the *Message Receiving Ratio* ψ of attackers has a higher value and varies in a wider range when the drop probability of attackers becomes greater. Keep in mind, a higher *Message Receiving Ratio* ψ denotes malicious behavior. In our system, when the attackers with the highest drop probability exist in the network, they have the strongest ability to destroy the network. When the drop probability of attackers is equal to 1, attackers obtain the messages and drop all of them, hence, the attackers

have the strongest ability to disturb the network. We can see in Figures 5.1 (a), 5.2 (a) and 5.3 (a), when blackhole attackers ($p = 1$) exist in the network, the values of attackers' *Message Receiving Ratio* ψ using Epidemic, Maxprop and PROPHET are highest. The reason is that the blackhole attackers keep on dropping the same messages receiving from different nodes, making the value of ψ very huge. When the drop probability varies from 0.9 to 0.5, the greyhole attackers' ability to destroy the network gradually decreases, hence, the attackers' *Message Receiving Ratio* ψ becomes smaller.

Using the unlimited-copy routing protocols, the *Message Forwarding Ratio* θ of attackers is distributed over a wider area as the drop probability of attackers decreases. As we mentioned before, a larger *Message Forwarding Ratio* θ indicates the cooperation of the nodes, while a smaller θ signifies the selfishness of nodes. When the blackhole attackers exist in the network, the values of the *Message Forwarding Ratio* θ are equal to 0 using different routing protocols in Figures 5.1 (a), 5.2 (a) and 5.3 (a), showing the non-cooperation of the blackhole attackers. When the drop probability varies from 0.9 to 0.5, the greyhole attackers partly transfer messages for other nodes. As seen in Figures 5.1 (f), 5.2 (f) and 5.3 (f), when the drop probability equals to 0.5, attackers have 50% chance to drop the messages and 50% chance to transfer the messages, hence, the *Message Forwarding Ratio* θ will increase as the attackers sometimes help other nodes transfer messages.

In Figure 5.1 to Figure 5.3, the *Message Receiving Ratio* ψ of normal nodes always remains unchanged. Normal nodes will not accept a message if they already have the message in their buffer, hence, the *Message Receiving Ratio* ψ of normal nodes is equal to 1. However, the *Message Forwarding Ratio* θ of normal nodes will vary over a wider range when attackers exist in the system. This is due to the fact when a normal node encounters an attacker with a high drop probability, the normal node will try to send all of the messages which are not in the buffer of the attacker, however, the attacker only has the messages generated by itself or very few messages to transfer back to the normal node. According to the definition of the *Message Forwarding Ratio* θ (see formula 5.1), more *forwarded* messages over less *received* messages leads to the bigger value of θ.

Figure 5.4 shows the behavior of nodes under the limited-copy routing protocol Spray and Wait. The *Message Forwarding Ratio* θ and the *Message Receiving Ratio* ψ also follow the principle we discuss above. However, compared to the unlimited-copy routing protocols, the attackers' *Message Receiving Ratio* ψ varies in a much lower and smaller range using Spray and Wait. With the limited number of replicas in the system, the total number of messages dropped by the attackers has an upper bound, hence, the ψ varies not as wide as the same condition of unlimited-copy routing protocols.

From Figure 5.1 to Figure 5.4, the results show that the (θ, ψ) tuple can clearly describe the different behavior between normal and malicious nodes. Hence, we will apply these two indicators to build our MDS.

5.2.2 Cluster Analysis

Cluster analysis is a popular approach to implement the partitioning operation and discover the natural grouping from a set of points or objects [115, 116, 117, 118]. There are many different types of cluster analysis but most share the same basic principle: On the basis of objects' similarity with respect to some defined criteria, cluster analysis forms groups in such a way that objects in the same group are similar to each other (high within-group homogeneity), whereas objects in different group are dissimilar as possible (low between-group homogeneity). Cluster analysis can finally partition objects with similar characteristics into the same cluster, making it a frequent classification tool. In our system, the behavior of malicious nodes and benign nodes exhibits significant differences (see section 5.2.1.1), making cluster analysis a very suitable mechanism to detect malicious nodes.

5.2.2.1 K-means Clustering

Due to its ease of implementation, performance and simplicity K-means clustering [116, 119, 120] became a well known and widely used algorithm for cluster analysis. We will use K-means clustering in our system. K-means clustering deals with a set of n points and clusters them into K groups based on a measure of similarity. Points with high similarities are assigned in the same group while the similarities among different groups are low. In K-means clustering, similarity metric, cluster initialization and the number of clusters K are the three user-specified parameters. We use 2-dimensional data-points (θ, ψ) as the input data, that characterize the behavior of a node (see section 5.2.1).

In K-means clustering, the choice of similarity measure can have a profound impact on the cluster analysis results, as it determines whether the classification is based on the characteristic pattern of objects. In our system, similarity is expressed by the Euclidean distance between points and cluster centers. A point (θ, ψ) is assigned to the cluster c_k whose cluster centroid μ_k has the greatest similarity (lowest distance) with the point. The goal of K-means clustering is to minimize the sum of the Euclidean distances between all points x_i and cluster centroids μ over all K clusters as shown in formula (5.4).

$$D(C) \;=\; \sum_{k=1}^{K} \sum_{x_i \in c_k} \sqrt{(x_{i\theta} \;-\; \mu_{k\theta})^2 \;+\; (x_{i\psi} \;-\; \mu_{k\psi})^2} \tag{5.4}$$

K-means clustering uses iterative partitioning technique. On a high level these are the steps of K-means algorithm:

1) Choose K points as initial centroids $\mu \;=\; \{\mu_k,\; k \in (1, ..., K)\}$.
2) Assign all points x_i to the nearest centroids μ.
3) Recalculate each centroid of the generated clusters.
4) Go to step 2. Break, if the algorithm converges ($D(C)$ is minimized) and centroids stabilize.

Using iterative partitioning method, K-means clustering begins by calculating the cluster centroids and locating the entities to their nearest cluster centroid. Then K-means clustering will continue the process of calculating the new cluster centroids and relocating entities to their new nearest cluster centroid until all the entities are stable and closest to their own cluster centroid. In our system, we allow the iteration process of calculating new centroids no more than 1000 times.

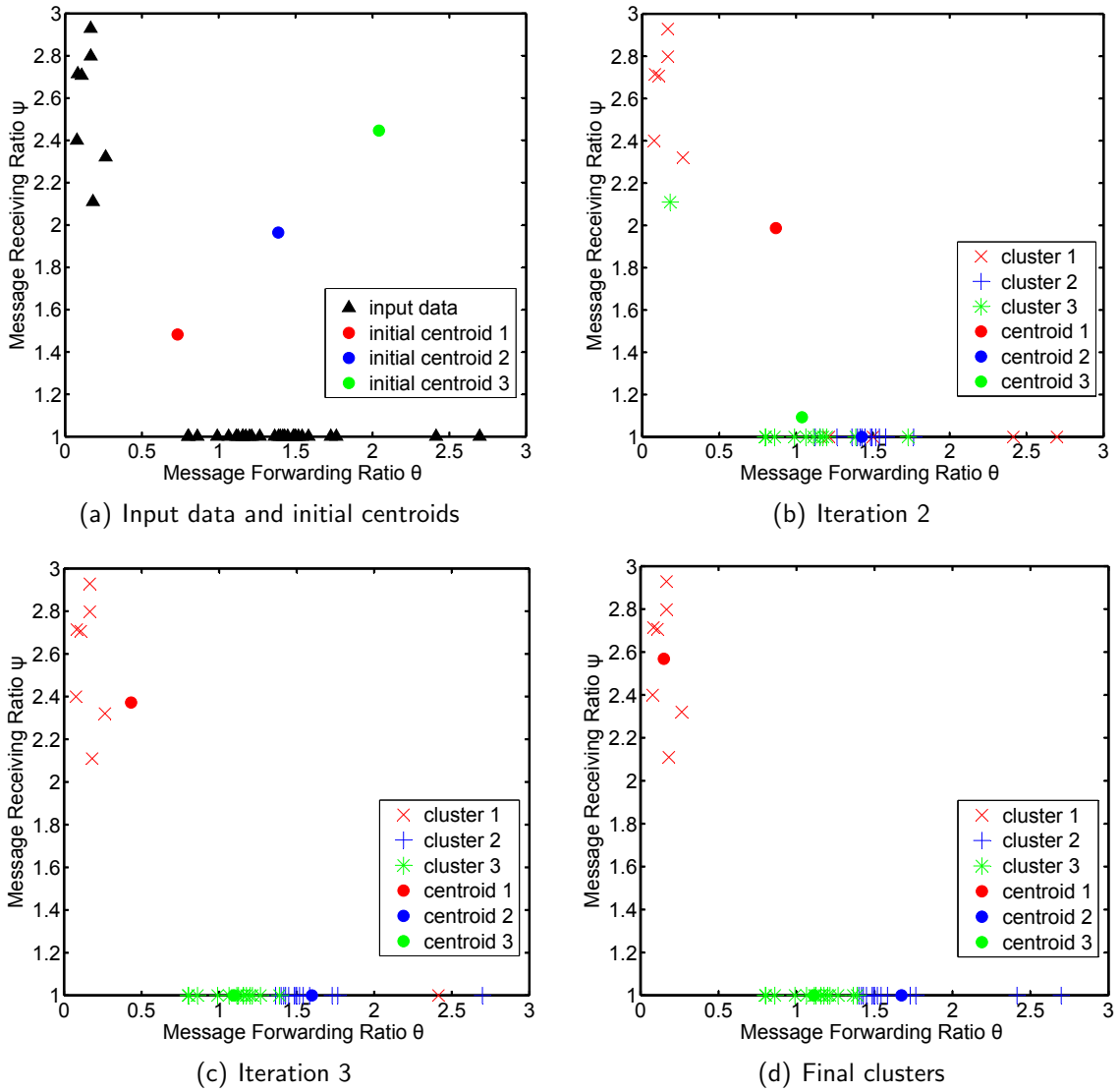
Obviously, the choice of different initial centroids can lead to different clusters. In our system the initial centroids are chosen as follows: For each dimension the bounding box of the area covered by the data points x_i is divided into slice of length s such that

$$s_\theta \;=\; \frac{x_{\theta max} \;-\; x_{\theta min}}{K} \quad \textbf{and} \quad s_\psi \;=\; \frac{x_{\psi max} \;-\; x_{\psi min}}{K} \tag{5.5}$$

The initial centroid coordinates μ_k will be placed into the center of each slice in each dimension as formula (5.6):

$$\begin{aligned} \mu_k \;=\; (x_{\theta min} \;+\; s_\theta(k \;-\; 0.5),\; x_{\psi min} \;+\; s_\psi(k \;-\; 0.5)) \\ k \in \{1, ..., K\} \end{aligned} \tag{5.6}$$

K-means clustering requires that the number of clusters need to be specified before the analysis takes place. However, determining the number of clusters is one of the most difficult problems in cluster analysis. Much work has been done in this area [121, 122, 123]. In K-means clustering, a more suitable value of K leads to more meaningful clusters.



(a) Input data and initial centroids

(b) Iteration 2

(c) Iteration 3

(d) Final clusters

Figure 5.5: **K-means Algorithm. (a) 2-dimensional input data-points and 3 initial centroids; (b)(c) choose centroids and cluster data-points according to formula (5.4); (d) obtains the final clusters.**

In our system, the number of cluster K is chosen according to the behavior of nodes. Using K-means clustering, our goal is to distinguish the behavior of normal nodes from the behavior of malicious nodes. However, we should also consider some special conditions: 1) When a node first joins into the network, it needs some time to adapt to the network, therefore, it may happen that the data-points (θ, ψ) about the node is not convictive to demonstrate its normal behavior. Hence, the K-means clustering needs to give these nodes some time to integrate into the network. 2) Different nodes have different opportunities to communicate with other nodes hence have various

characteristics of the behavior. Active nodes can show their behavior well because more opportunities connecting with other nodes provide better data-points (θ, ψ). However, there also exists the nodes which cannot make a greater contribution to the network due to the scarcer communication opportunities. Therefore, our system should give tolerance to the different behavior from the normal nodes. Considering different perspectives, we choose K to be 3. One cluster is responsible for collecting malicious nodes, we use the second cluster to give tolerance to the system due to the variance of vehicular nodes in the network, and we design the third cluster to gather normal behavior. The detail will be introduced in section 5.2.4. Figure 5.5 shows how the K-means clustering works.

5.2.3 Improved TR Update Principle

As noted in section 2.4.3, the node using TFT will initially cooperate and then does what the other counterpart did in the last round, in order to induce its counterpart to cooperate. According to paper [84], TFT is successful because of its three strategies: The nice strategy, the provocable strategy and the forgiving strategy. The nice strategy requires that nodes will not first defect. When both encountering nodes adopt the nice strategy, they can well cooperate with each other. The provocable strategy responds when detecting the defection of encountering node, the node will also defect. A forgiving strategy is adopted when its opponent returns to cooperation, the node will follow. TFT is a simple and effective behavioral rule for a variety of node interactions, that links incentive mechanisms with reputation. Hence, in our system our TR update scheme enforces a TFT principle.

We define three strategies – the unforgiving strategy, the forgiving strategy and the nice strategy. We will use these strategies to modify the additive component λ and the subtractive component γ during consecutive detections. Our mechanism for cooperation has been proposed as follows (see Figure 5.6): Configuration 1: If the TR of the node has been decreased during the previous encounter and the node is classified as malicious again, we will adopt an unforgiving strategy and double the decreasing component γ. Configuration 2: However, if a previously suspected node behaves well this time, the judging node will use a forgiving strategy and increase the node's TR as usual. Configuration 3: If a node's TR was increased during the previous encounter, and this time the node is detected as benign again, the judging node will reward good behavior and adopt a nice strategy by doubling λ. Configuration 4: If this time the node is classified as malicious, a forgiving strategy will be adopted that the TR will decrease as usual. In the first round of the encounter, nodes will always adopt a forgiving strategy. We incorporate the idea of the TFT principle into the TR Update (see section 5.2.4) to influence the behavior of other nodes. In our system we choose $T_{initial} = 0.5$, $T_{evil} = 0.3$, $T_{friend} = 0.8$, $\gamma = 0.04$ and $\lambda = 0.04$.

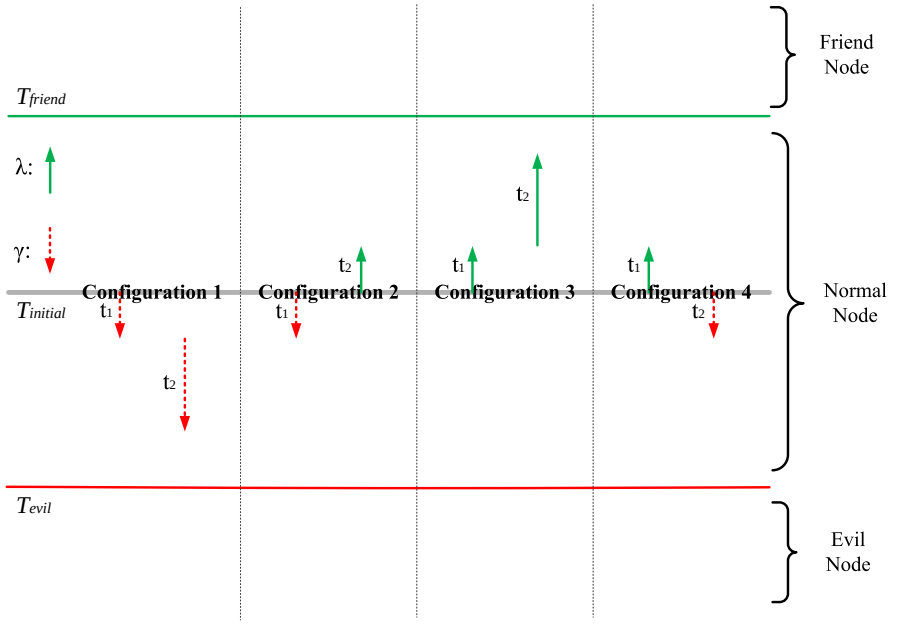


Figure 5.6: Using Basic Principle of TFT to Update TR

5.2.4 TR Update

After calculating (θ, ψ) of node j, node i will retrieve (θ, ψ) tuples of other nodes from its ML and perform K-means clustering with $K = 3$ if enough (> 9) (θ, ψ) pairs are available. Otherwise only the updated (θ, ψ) for node j will be added to i's ML.

Three groups generated by K-means clustering are assigned to the three classes *Suspected*, *Unsure* and *Trusted*. Nodes in the *Suspected* group have the highest probability of being malicious, nodes in the *Trusted* group the lowest. The groups are assigned to the three classes based on their average *Message Receiving Ratio* ψ_{Avg}. Since malicious nodes will exhibit higher ψ values, the group with the largest ψ_{Avg} is used as the *Suspected* group, and the one with the lowest ψ_{Avg} as the *Trusted* group. In case of equal ψ_{Avg} values the respective average *Message Forwarding Ratio* θ_{Avg} are used as a fallback, where smaller θ values indicate less cooperative nodes.

The TR update principle is that the system makes sure to punish or reward nodes that clearly exhibit malicious or exemplary behavior by modifying their TR ratings. The actions of nodes in different groups are weighted differently (see Algorithm 3): If node j belongs to the *Suspected* group it receives the hardest punishment for bad behavior. The best a node in the *Suspected* group can hope for, is no punishment. If $\psi_j > \psi_{opt}$, TR_i^j will be decreased according to formula (5.7) of $k = 1$. According to the TFT principle in section 5.2.3, only if node j was malicious last time, then

the system will double k, and therefore set $k = 2$. If $\psi_j = \psi_{opt} = 1$, θ_j is taken into consideration. If $\theta_j < \theta_{SuspectedAvg}$ it seems j behaves comparatively worse than other nodes in its cluster. In this case, in light of the acceptable ψ_j, j will only be punished according to formula (5.7) of $k = 0.5$. Considering the TFT principle, only if node j behaved maliciously last time, then we will double k and set $k = 1$. If $\theta_j \geq \theta_{SuspectedAvg}$, j will receive no punishment and TR_i^j will not be changed.

$$TR_i^j \;=\; TR_i^j \;-\; k \cdot \gamma \quad (0 < \gamma < 1,\; k \in \{0.5,\; 1,\; 2\}) \tag{5.7}$$

Nodes in the *Unsure* group have the chance to get a slight punishment or a slight encouragement. In this group if $\psi_j > \psi_{opt}$ and $\theta_j < \theta_{UnsureAvg}$, too much evidence shows j's malicious behavior, hence, the TR will be decreased according to formula (5.7) of $k = 1$. $k = 2$ will be adopted when node j was malicious before. If $\psi_j > \psi_{opt}$ and $\theta_j \geq \theta_{UnsureAvg}$, $k = 0.5$ is used to decrease the TR. We will choose $k = 1$ if it is under the unforgiving strategy of the TFT principle. If $\psi_j = \psi_{opt}$, two cases are considered: If $\theta_j < \theta_{UnsureAvg}$ the TR will not be changed. Else, if $\theta_j \geq \theta_{UnsureAvg}$ j will be encouraged according to formula (5.8) of $m = 0.5$. If j behaves benignly last time, m will be doubled and set $m = 1$ (see section 5.2.3).

$$TR_i^j \;=\; TR_i^j \;+\; m \cdot \lambda \quad (0 < \lambda < 1,\; m \in \{0.5,\; 1,\; 2\}) \tag{5.8}$$

A node in the *Trusted* group has the chance to get the largest boost for its TR. If $\psi_j = \psi_{opt}$, its TR will be increased according to formula (5.8) of $m = 1$. Only when node j was rewarded last time, then the system can double m and set $m = 2$. When $\psi_j > \psi_{opt}$ the TR will not be changed, to prevent the condition that node i is surrounded by malicious nodes and keeps increasing their TRs.

Apart from the rules outlined above, a node will keep a record about previous classified nodes in its ML and then next time use TFT as described before to update the TR.

Algorithm 3: **Update TRs**

i will update TR_i^j for j based on ER_j received from Algorithm 1 (see *Rule Violation Checks* in section 3.3.1)

$dataPoints$.ADD(θ_j, ψ_j, j)
$dataPoints$.ADD(ML_i)
($v_{Suspected}$, v_{Unsure}, $v_{Trusted}$) = GETKMEANSOUTPUT($dataPoints$)
$Avg_{\theta S}$ = GETFORWARDINGRATIOAVERAGE($v_{Suspected}$)
$Avg_{\theta U}$ = GETFORWARDINGRATIOAVERAGE(v_{Unsure})
$Avg_{\theta T}$ = GETFORWARDINGRATIOAVERAGE($v_{Trusted}$)
$misbehaving$ = GETISMISBEHAVING(ML_i, j)
$\psi_{opt} = 1$
if $j \in v_{Suspected}$ **then**
    if $\psi_j > \psi_{opt}$ **and not** $misbehaving$ **then**
        $TR_i^j = TR_i{}^j - \gamma$
    else if $\psi_j > \psi_{opt}$ **and** $misbehaving$ **then**
        $TR_i^j = TR_i{}^j - 2 * \gamma$
    else if $\psi_j == \psi_{opt}$ **and** $\theta_j < Avg_{\theta S}$ **and not** $misbehaving$ **then**
        $TR_i^j = TR_i{}^j - \gamma/2$
    else if $\psi_j == \psi_{opt}$ **and** $\theta_j < Avg_{\theta S}$ **and** $misbehaving$ **then**
        $TR_i^j = TR_i{}^j - \gamma$
    else
        $TR_i^j = TR_i{}^j$
    end if
else if $j \in v_{Unsure}$ **then**
    if $\psi_j > \psi_{opt}$ **and** $\theta_j < Avg_{\theta U}$ **and not** $misbehaving$ **then**
        $TR_i^j = TR_i{}^j - \gamma$
    else if $\psi_j > \psi_{opt}$ **and** $\theta_j < Avg_{\theta U}$ **and** $misbehaving$ **then**
        $TR_i^j = TR_i{}^j - 2 * \gamma$
    else if $\psi_j > \psi_{opt}$ **and** ($\theta_j > Avg_{\theta U}$ **or** $\theta_j == Avg_{\theta U}$) **and not** $misbehaving$ **then**
        $TR_i^j = TR_i{}^j - \gamma/2$
    else if $\psi_j > \psi_{opt}$ **and** ($\theta_j > Avg_{\theta U}$ **or** $\theta_j == Avg_{\theta U}$) **and** $misbehaving$ **then**
        $TR_i^j = TR_i{}^j - \gamma$
    else if $\psi_j == \psi_{opt}$ **and** $\theta_j < Avg_{\theta U}$ **then**
        $TR_i^j = TR_i{}^j$
    else if $\psi_j == \psi_{opt}$ **and** ($\theta_j > Avg_{\theta U}$ **or** $\theta_j == Avg_{\theta U}$) **and** $misbehaving$ **then**
        $TR_i^j = TR_i{}^j + \lambda/2$
    else
        $TR_i^j = TR_i{}^j + \lambda$
    end if
else
    if $\psi_j > \psi_{opt}$ **then**
        $TR_i^j = TR_i{}^j$
    else if $\psi_j == \psi_{opt}$ **and** $misbehaving$ **then**
        $TR_i^j = TR_i{}^j + \lambda$
    else
        $TR_i^j = TR_i{}^j + 2 * \lambda$
    end if
end if
TR_i^j = MAX(TR_i^j, 1.0)
TR_i^j = MIN(TR_i^j, 0.0)

5.3 Performance Evaluations in Homogenous VDTNs

In this section, we will evaluate whether our proposed cluster analysis based MDS is able to defend the VDTNs against attackers. The vehicular network consists of the homogenous vehicular nodes. Vehicular nodes will use cluster analysis to distinguish the behavior of encountered nodes and then build a reputation system to identify malicious nodes. We evaluate our method in different VDTN scenarios using different DTN routing protocols.

5.3.1 Simulation Setup

We use *The ONE* DTN simulator to evaluate our MDS (see section 3.4.1). A comparatively dense Helsinki scenario using a synthetic movement model, a sparse San Francisco scenario using GPS traces and a Braunschweig scenario using the public transportation vehicles to collect air pollution information are used for evaluations. We perform comparative measurements using the Epidemic, MaxProp, PROPHET and Spray and Wait routing protocols. The basic parameters for these three scenarios are given below.

5.3.1.1 Helsinki Scenario

We use the same parameters in the Helsinki scenario as introduced in section 4.3.1. 40 vehicular nodes spread all over Helsinki, with a transmission radius of 200 meters and a buffer size of 1 GB. 4, 8 and 12 nodes (10%, 20%, 30%) among the 40 nodes are randomly chosen as malicious nodes whose drop probability varies from 0.5 to 1 and these malicious nodes will independently attack the system or collude with each other to invade the system. For detail information refers to Table 4.1.

Unless otherwise noted the results presented for each scenario are the average, min and max results of 10 simulation runs. Vehicular nodes apply the MDS starting from second 5000.

5.3.1.2 San Francisco Scenario

San Francisco scenario [108] is a real trace-driven simulation based on San Francisco GPS data. The data set contains mobility traces of approximately 500 taxi cabs driving in San Francisco over 30 days. We restrict our analysis to the 122 taxi cabs with mobility traces longer than 25000 GPS readings and running in a map size of 20000 m × 20000 m. A randomly chosen source node will generate one message to a randomly chosen destination in every 8 to 11 second. Each vehicular node will own a 1 GB buffer size and a transmission radius of 200 meters. We run the simulation for 24 hours (86 400 s). 12, 24 and 36 nodes (10%, 20%, 30%) are randomly chosen among the 122 nodes as malicious nodes whose drop probability varies from 0.5 to 1. These malicious nodes will independently attack the system and also have the ability to collude with each other as described before. Table 5.1 describes the basic parameters for the San Francisco scenario. Unless otherwise noted, the simulation results presented for each scenario are the average results of 5 experimental runs. Nodes apply the MDS starting from second 5000.

Table 5.1: **Simulation Parameters in the San Francisco Scenario**

Value	**San Francisco**
# nodes	122
Area	20000 m × 20000 m
Transmission radius	200 m
Movement	GPS trace
Buffer size	1 GB
Simulation time	24 h
Malicious nodes	10%, 20%, 30%
Drop probabilities	0.5-1.0

5.3.1.3 Braunschweig Scenario

In Braunschweig network [38], buses and trams move in the whole metropolitan area, collect the air pollution measurements and send the data to the gateway. The gateway is responsible for gathering messages and sends messages to the data center to analyze. Although buses and trams run on fixed paths, traffic congestion and vehicle dispatching make each contact of buses and trams unpredictable. Hence, buses and trams should not waste any contact opportunity to forward their messages.

Braunschweig scenario is a simulated scenario based on real world context data. By integrating information of the roads, lines, timetables and stops, the scenario

has one gateway, 54 stops and 28 buses and trams which are simultaneously running on 13 assigned lines. In an interval between 25 and 30 seconds, a node which is randomly chosen generates one message to the gateway. The gateway located near the Braunschweig main station receives messages and sends them to the data center. The lifetime of each message is 45 minutes and the size of messages is between 500 kB and 1 MB. Each node owns a 1 GB buffer size and a transmission radius of 200 meters. We randomly choose 3, 6 and 9 nodes (10%, 20%, 30%) among the 28 buses and trams as malicious nodes whose drop probability varies from 0.5 to 1 to independently or collusively attack the system. Table 5.2 shows the basic parameters of the Braunschweig scenario.

We run the simulation for 12 hours (43 200 s). In the first hour buses and trams are successively integrated into the simulation which takes some time until the network is fully populated. For this reason buses and trams apply the MDS starting from second 5000. Unless otherwise noted, the simulation results presented for each scenario are the average, maximum and minimum results of 10 experimental runs.

Table 5.2: Simulation Parameters in the Braunschweig Scenario

Value	Braunschweig
# nodes	28 nodes + 1 gateway
Area	8899 m × 8096 m
Transmission radius	200 m
Movement	simulated movements based on real world context data
Buffer size	1 GB
Simulation time	12 h
Malicious nodes	10%, 20%, 30%
Drop probabilities	0.5-1.0

5.3.2 Detection Rate

We studied the detection rates versus different number of malicious nodes with different drop probabilities using the same four different routing protocols. To better understand the performance of our MDS the average and the maximum and minimum detection rates are shown in Figures 5.7 to 5.9.

Any malicious node should be purged from the network as fast as possible. The results depicted in Figure 5.7, Figure 5.8 and Figure 5.9 show that, detecting malicious nodes with a high drop probability, which have a more severe impact on the network

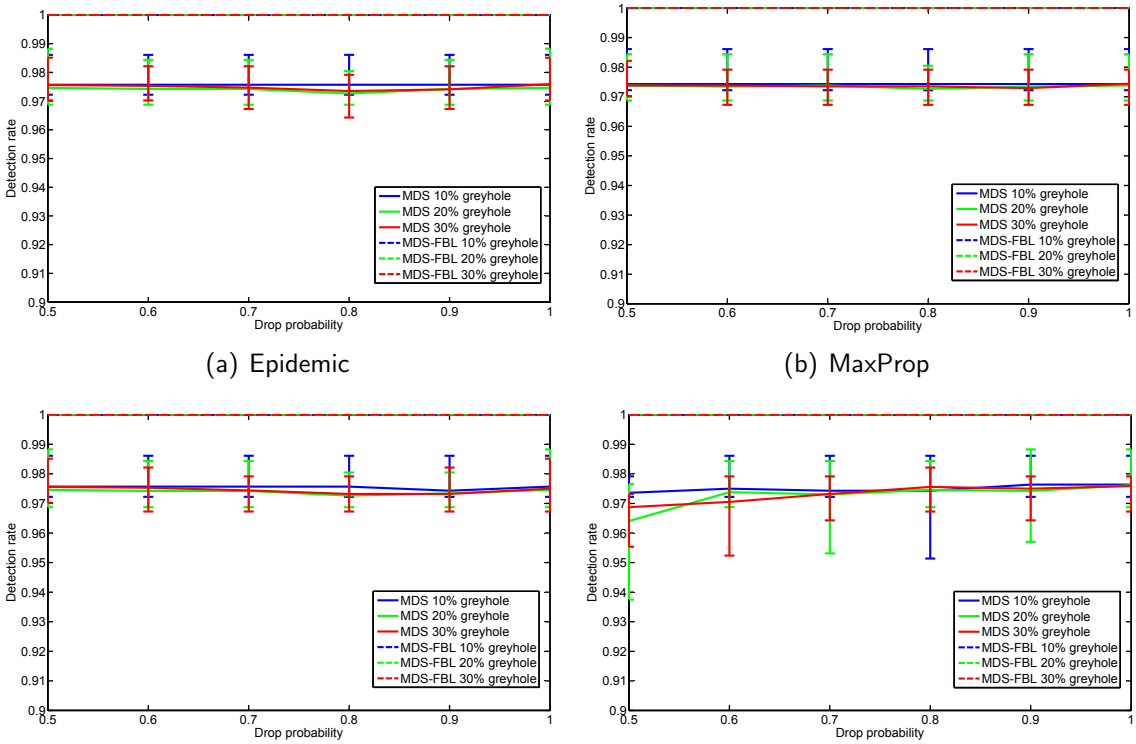
performance, our MDS can achieve a higher detection rate. For the malicious nodes with a low drop probability that have a less pronounced effect on the network performance, our MDS can keep a comparably high detection rate. Under the current simulation time, the detection rate of evil nodes with a lower drop probability is not as high as the detection rate of evil nodes with a higher drop probability. This is due to the fact that the attackers with low drop probabilities have similar behavior to normal nodes, this increases the difficulty to detect malicious nodes, hence, the detection rates slightly decrease.

In addition, when the FBL extension is used, our MDS consistently achieves a much higher detection rate no matter that the system varies different scenarios using different routing protocols or it consists of different number of blackhole or greyhole attackers. The FBL mechanism can help friend nodes make a final decision considering friends' suggestion, detecting suspected nodes faster under the limited time.

In these three scenarios, for Spray and Wait in Figures 5.7 (d), 5.8 (d) and 5.9 (d), the detection rate of malicious nodes is not as high as the detection rate using the unlimited-copy routing protocols. The reason is that due to much less messages transferred using Spray and Wait, our MDS will obtain limited information provided by ERs. Compared to the unlimited-copy routing protocols, the decision of the MDS under Spray and Wait will be based on a much smaller number of ERs, hence, it increases the difficultly for our MDS to make a correct decision at once.

As seen in Figure 5.7, dealing with blackhole attackers ($p = 1$), our system performs well and can achieve a high average detection rate up to 97.6%, 97.4%, 97.6% and 97.6% using four different routing protocols Epidemic, MaxProp, PROPHET and Spray and Wait in the Helsinki scenario. Compared to the detection rate up to 97.4%, 97.4%, 97.3% and 97.4% using the adaptive threshold-based MDS in Figure 4.5, both MDSs keep a good detection rate. The results depicted in Figures 5.7 (a) to 5.7 (c) show that, when greyhole attackers adopt different drop probabilities ($0.5 \leq p \leq 0.9$), our MDS can sustain a comparably average high detection rate ranging from 97.3% to 97.6%, 97.3% to 97.4% and 97.3% to 97.6% in Epidemic, MaxProp and PROPHET. Coping with greyhole attackers, our cluster analysis based MDS achieves a better detection rate compared to the detection rate of 93.9% to 97.2%, 96.9% to 97.1% and 96.5% to 97.2% using the adaptive threshold-based MDS (Figure 4.5) in the same routing protocols. Using Spray and Wait our MDS achieves a high detection rate coping with blackhole attackers. However, in Figure 5.7 (d) with drop probabilities decreasing, the detection rates decrease a bit. We can see when there are 30% greyhole attackers adopting the drop probability of 0.5, the detection rate achieves 96.9% in Figure 5.7 (d). These detection rates are improved significantly compared to the detection rates using the adaptive threshold-based MDS. In our simulations we also evaluate our MDS using the FBL principle. When adding the FBL our MDS

can achieve 100% detection rate in Epidemic, MaxProp, PROPHET and Spray and Wait.



(a) Epidemic (b) MaxProp

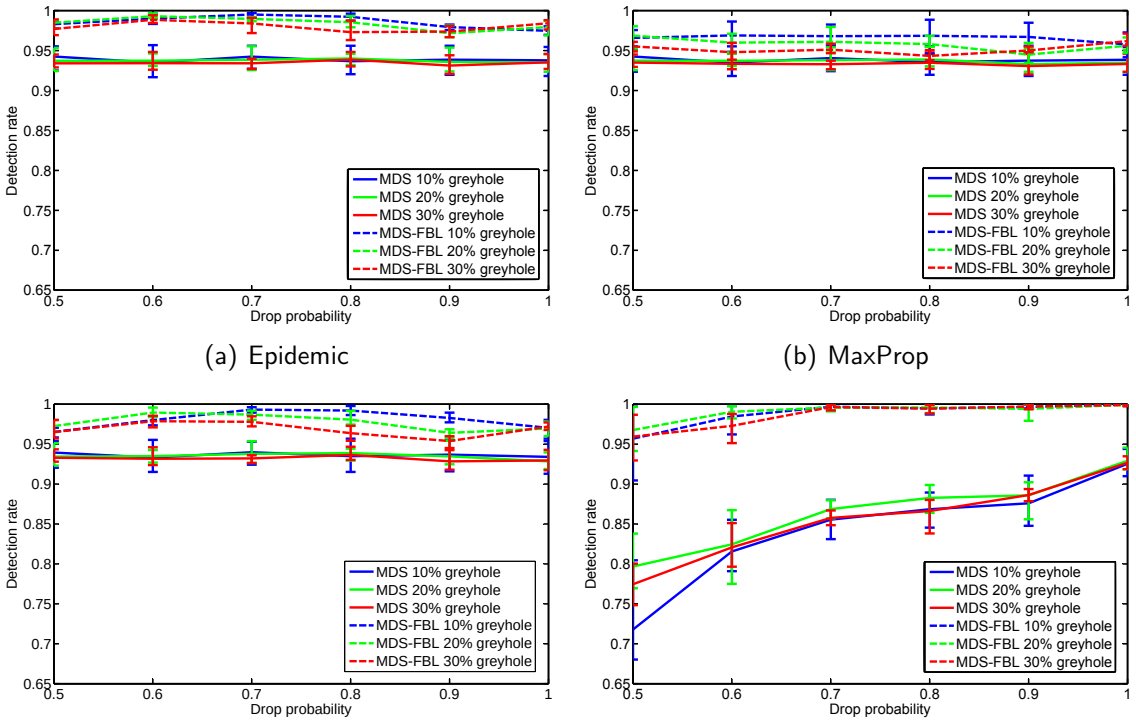
(c) PROPHET (d) Spray and Wait

Figure 5.7: **Detection Rate – Using Cluster Analysis in the Helsinki Scenario**

Compared to the Helsinki scenario, in the San Francisco scenario the vehicular nodes run in a comparably sparse condition. The distances between vehicular nodes are far larger and the number of encounters is limited. Therefore, the San Francisco detection rates are a bit lower than the Helsinki detection rates. However, when blackhole nodes attack the system, our MDS can achieve a high average detection rate up to 93.8%, 93.8%, 93.4% and 92.9% under different routing protocols. As greyhole attackers vary their drop probabilities, our MDS can keep a stable and high average detection rate of more than 93.1%, 93.1% and 92.8% using Epidemic, MaxProp and PROPHET in Figures 5.8 (a) to 5.8 (c). Under the current simulation time, using Spray and Wait (see Figure 5.8 (d)), the detection rate of malicious nodes with a lower drop probability is not as high as the detection rate of malicious nodes with a higher drop probability.

Using the FBL mechanism, the average detection rates of our system increase in all cases. Especially for Spray and Wait, the average detection rates by using the FBL

mechanism improve from 7% to 23.9% when attackers adopt different drop probabilities. The results show that the FBL mechanism can help nodes share information about the system and improve the system performance under the limited simulation time.



(a) Epidemic (b) MaxProp

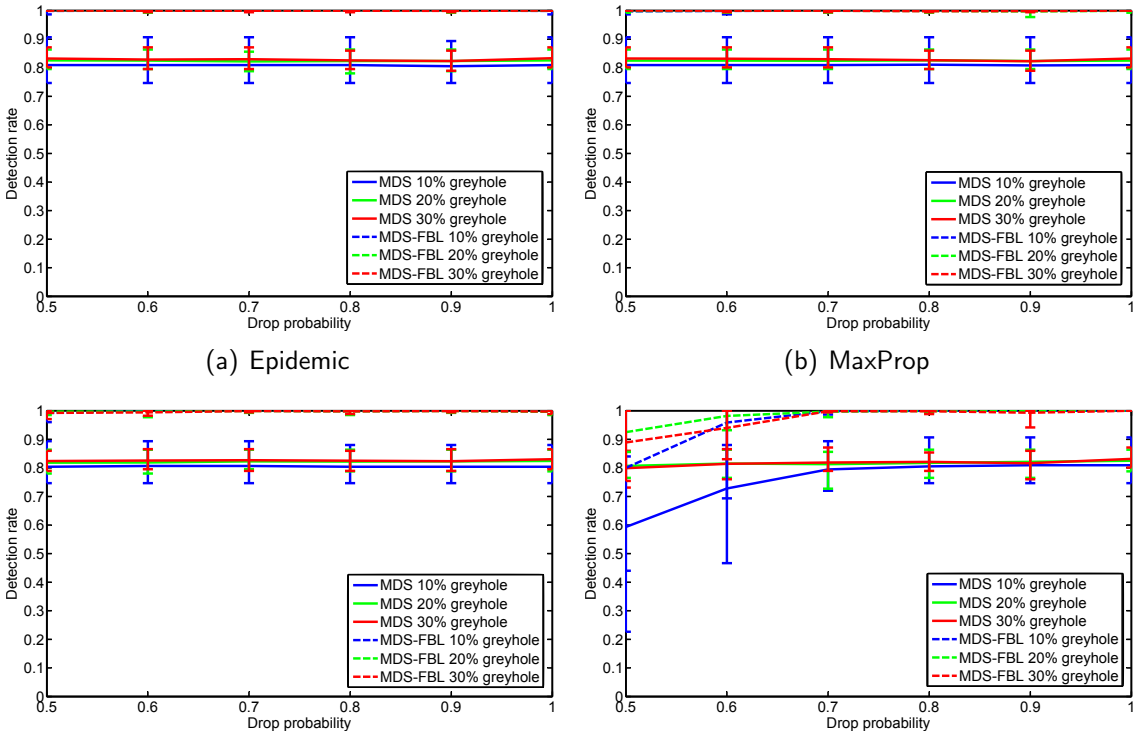
(c) PROPHET (d) Spray and Wait

Figure 5.8: **Detection Rate – Using Cluster Analysis in the San Francisco Scenario**

The results depicted in Figure 5.9 show that, in the Braunschweig scenario, when greyhole attackers adopt different drop probabilities, our MDS can sustain an average detection rate of more than 80.5%, 80.8% and 80.4% in Epidemic, MaxProp and PROPHET respectively. Applying the FBL mechanism, our MDS can achieve nearly 100% detection rate using the same routing protocols. Using Spray and Wait in Figure 5.9 (d), when greyhole attackers adopt a high drop probability ($p \geq 0.7$), the MDS can keep the average detection rate bigger than 79.5%. However, with lower drop probabilities, the detection rates decrease a bit. The FBL mechanism helps the detection rate under Spray and Wait achieve 100% when attackers use a high drop probability ($p \geq 0.7$) and the detection rate is in general also good when attackers adopt a low drop probability.

In this thesis, we use the strict definition for the detection rate here: A malicious node needs to be detected by *all* other nodes. However, in the San Francisco and Braunschweig scenario, some of the vehicular nodes never meet each other, hence,

definitely the detection rate cannot achieve 100% without using the FBL mechanism. Even for lower overall detection rates according to this definition, vehicular nodes which are more frequently near a malicious node have a higher chance detecting it. Our extensive simulations demonstrate our MDS can efficiently detect evil nodes with varying drop probabilities under different scenarios, yielding a high detection rate in the system. Meanwhile, with the FBL mechanism, our MDS can achieve a better performance.



(a) Epidemic (b) MaxProp

(c) PROPHET (d) Spray and Wait

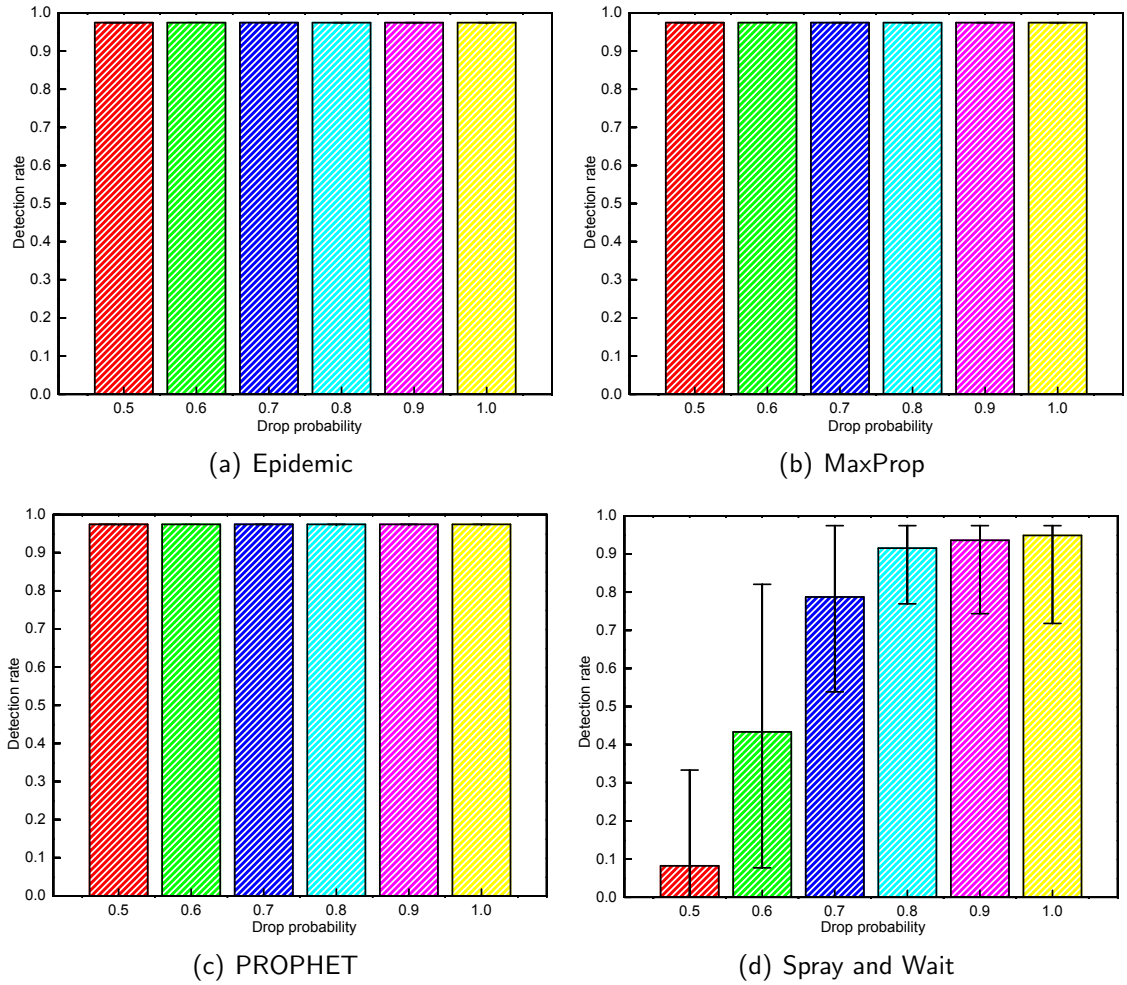
Figure 5.9: **Detection Rate – Using Cluster Analysis in the Braunschweig Scenario**

5.3.3 Detection Rate with Low Amount of Attackers

In section 5.3.2, we demonstrated when there are 10%, 20% and 30% attackers in the system, our MDS can keep high detection rates in different scenarios. The question we are interested in this part is, whether our approach is still suitable when there are even fewer malicious nodes in the system. To better understand the performance of the clustering algorithm, we choose only one attacker from the network, providing the clustering algorithm with most limited information about the malicious group, and test whether our MDS still performs well. In this section, we will focus on how

our MDS works when there is limited information about attackers available, hence, we disable the FBL mechanism.

In the Helsinki scenario, we use one given scenario with the same movement pattern and the same message generation mechanism to test our MDS coping with the condition when there is only one attacker in the network. The average, maximum and minimum detection rate results are obtained from 10 runs where one vehicular node is randomly chosen from 40 vehicular nodes as the malicious node. The results are shown in Figure 5.10.



(a) Epidemic (b) MaxProp

(c) PROPHET (d) Spray and Wait

Figure 5.10: Detection Rate with a Single Attacker – Using Cluster Analysis in the Helsinki Scenario

From Figures 5.10 (a) to 5.10 (c), we see that when there is only one attacker in the Helsinki scenario, our MDS can still sustain an average detection rate of 97.4%, 97.4% and 97.4% using Epidemic, MaxProp and PROPHET respectively. These high detection rates are obtained because the Helsinki scenario provides a dense vehicular

network. Vehicles have more opportunities to meet each other and more messages to transfer between each other, helping the clustering algorithm better distinguish nodes' behavior. For Spray and Wait in Figure 5.10 (d), our MDS can achieve a high detection rate when detecting malicious nodes with a high drop probability. With drop probabilities decreasing, the detection rate follows to decrease. The reason is that compared to the unlimited-copy routing protocols, Spray and Wait transfers less messages in the network and therefore provides the MDS with a much smaller number of ERs, increasing the difficultly for our MDS to make a correct decision especially when the behavior between normal nodes and attackers is similar.

We also evaluate our MDS coping with one single attacker in the San Francisco scenario. The movement of the 122 vehicular nodes in the San Francisco scenario is based on taxi GPS trace data. We first observe the behavior of the vehicular nodes and choose the top 61 (50%) vehicular nodes which have the most contact opportunities. We define that these top 61 vehicular nodes belong to the active group. Afterwards, using the same message generation mechanism, in each of the sub-runs, the system will randomly choose one attacker from the active group. The average detection rate is obtained from 10 runs.

As seen in Figures 5.11 (a) to 5.11 (c), dealing with one blackhole attacker ($p = 1$) in the network, our system can achieve an average detection rate of 52.6%, 92.8% and 57.8% using the routing protocols Epidemic, MaxProp and PROPHET in the San Francisco scenario. However, decreasing drop probabilities, the detection rates follow to decrease. Compared to the Helsinki scenario, the San Francisco scenario is a wider and sparser real world scenario. The large distances between vehicular nodes and the limited number of encounters increase the difficulty for vehicular nodes to judge the behavior of the other nodes. The detection rate of malicious nodes using Spray and Wait is not as high as the detection rate using the unlimited-copy routing protocols. The reason is that due to much less messages transferred using Spray and Wait, limited information can be provided by ERs, increasing the difficulty for our MDS to work efficiently.

In the Braunschweig scenario, there are 28 vehicular nodes which move according to the simulated movements based on real world context data. In the following sub-runs, we apply the same message generation pattern. Under the same drop probability, each of the 28 vehicular nodes will be chosen to work as an attacker. The average and the maximum and minimum detection rates are shown in Figure 5.12.

The results depicted in Figure 5.12 show that, in the Braunschweig scenario, even though there is only one attacker in the system, when the attacker adopts different drop probabilities, our MDS can still sustain an average detection rate of 82.2%, 82.2%, 82.0% and 77.8% using Epidemic, MaxProp, PROPHET and Spray and Wait

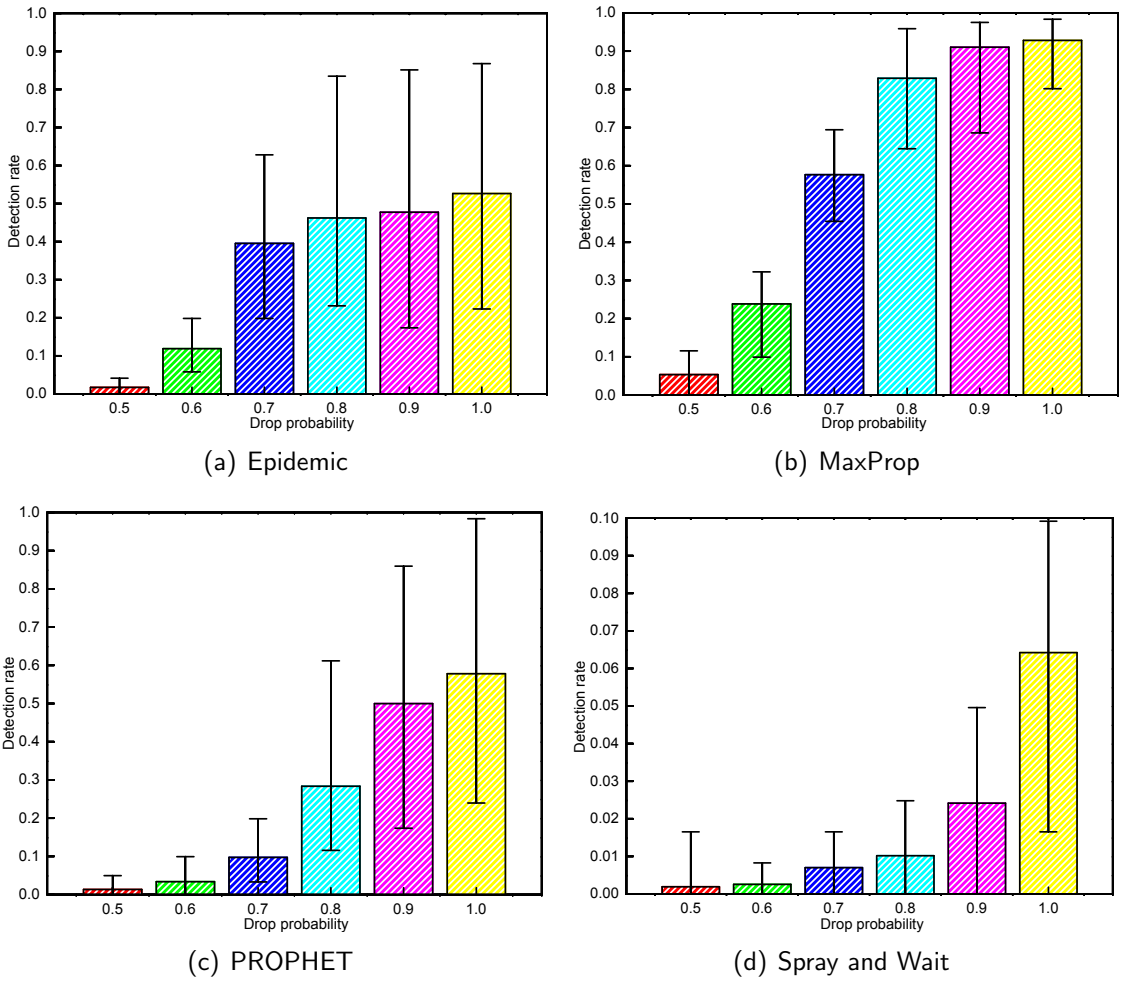
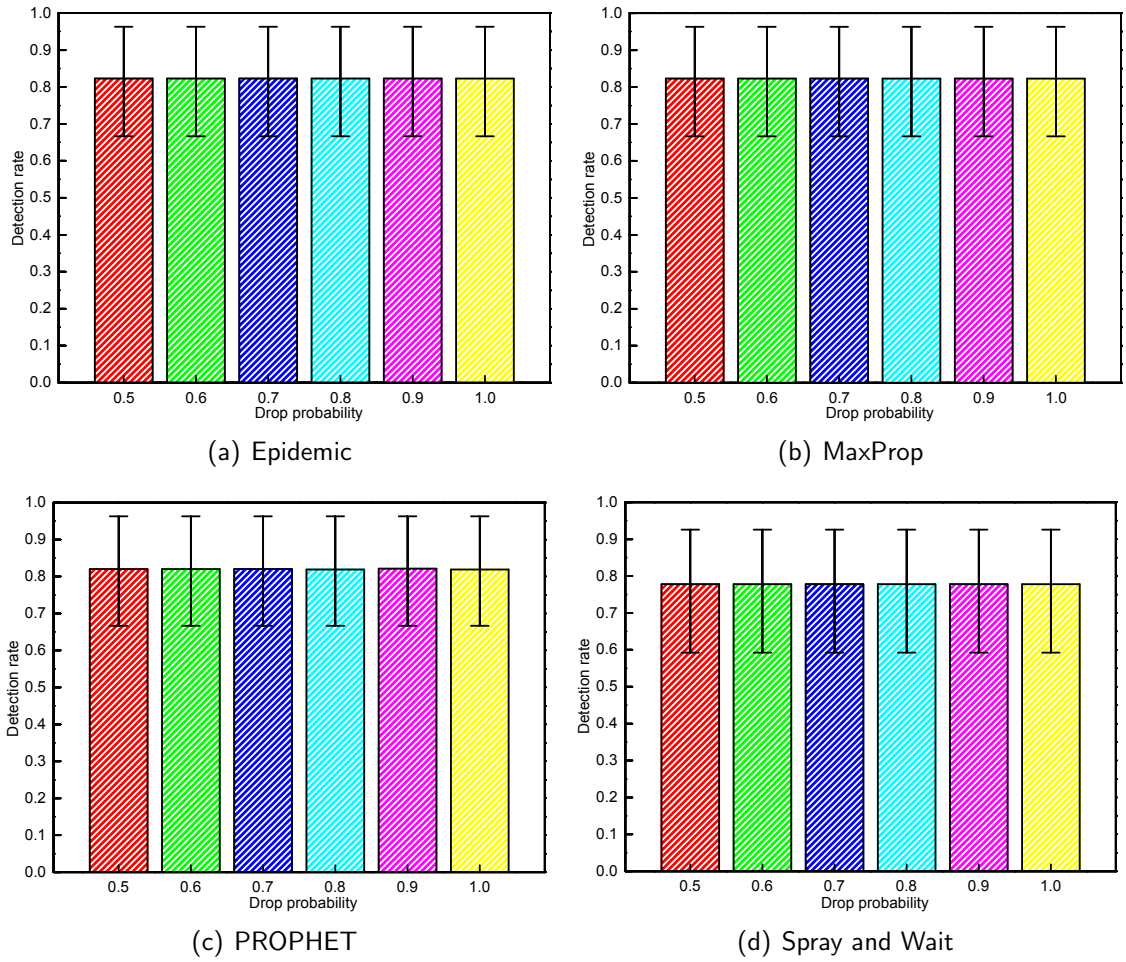


Figure 5.11: Detection Rate with a Single Attacker – Using Cluster Analysis in the San Francisco Scenario

respectively. In the Braunschweig scenario, because of the repeating behavioral patterns of buses and trams, our MDS has more opportunities to encounter other nodes and therefore can accurately distinguish the nodes' behavior.

Overall, we find that even if malicious nodes are the minority, and there might only be a single malicious node in the system, our MDS can still efficiently detect malicious nodes with varying drop probabilities under different scenarios. Besides, the more information ERs can provide for our MDS, the more accurate judgement our MDS can make.



(a) Epidemic (b) MaxProp

(c) PROPHET (d) Spray and Wait

Figure 5.12: Detection Rate with a Single Attacker – Using Cluster Analysis in the Braunschweig Scenario

5.3.4 Detection Rate with Colluding Attackers

The question we are interested in this section is, whether our MDS is still efficient when there are colluding attackers in the network. Figure 5.13 shows the average, maximum and minimum detection rates in three different scenarios with 10% to 30% blackhole attackers. Once these blackhole attackers are in each other's communication range, they will collude and generate bogus ERs (see *Behavior 9* in section 3.2.2.2). In Figure 5.13, "MDS" denotes the condition when the blackhole nodes independently attack the system. "MDS-CO" denotes the condition when blackhole attackers collude to generate bogus ERs with other attackers in their communication ranges. In this section, we do not evaluate the condition when these colluding nodes use the idea of the colluding group to attack the system (see *Behavior 10* in section 3.2.2.2), because this strategy has a high risk to be detected by the *Rule Violation*

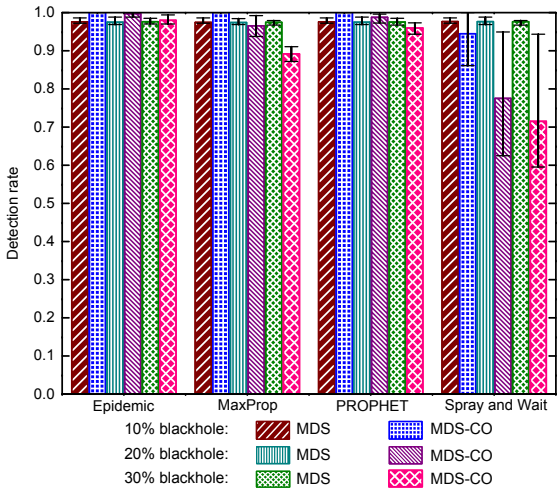
Checks as the results shown in section 4.3.3.2.

Figure 5.13 (a), Figure 5.13 (b) and Figure 5.13 (c) respectively show the detection rates in the Helsinki, San Francisco and Braunschweig scenario. Although the attackers forge beneficial ERs to cheat other normal nodes, for the unlimited-copy routing protocols such as Epidemic, MaxProp and PROPHET, our MDS can still sustain a comparably high detection rate compared to the condition when malicious nodes independently attack the system. However, as the number of colluding attackers increases, the detection rates also follow to go down a few percent. When there are 10%, 20% and 30% colluding blackhole attackers in the network, our MDS can obtain the detection rate of 100%, 99.7% and 98.0% in the Helsinki scenario, 94.4%, 90.4% and 83.8% in the San Francisco scenario and 81.9%, 83.0% and 78.8% in the Braunschweig scenario using the routing protocol Epidemic. Due to a large amount of transferred messages, enough information is provided by the ERs. Our MDS can defend against these colluding attackers and achieve a high detection rate. However, as more attackers collude with each other and more bogus ERs are generated, the collusion between malicious nodes will affect the judgement of the MDS and finally decrease the detection rates.

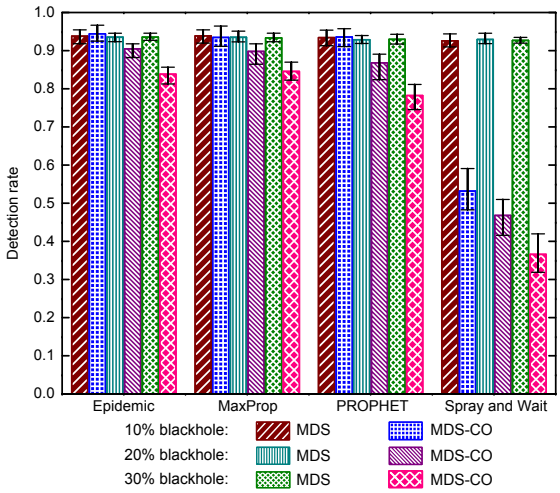
Normally colluding attackers increase the difficulty for the MDS to detect them. However, in Figure 5.13 (a), Figure 5.13 (b) and Figure 5.13 (c), interestingly, for the 10% blackhole case the tendency is reversed: Here the detection rate is actually improved when attackers collude. Because when a small amount of colluding attackers exists in the network, their bogus ERs only slightly improve their update ratio – (θ, ψ) tuples. These (θ, ψ) tuples are not good enough to completely hide their malicious behavior but on the contrary help other normal nodes adjust the (θ, ψ) boundary and make a more accurate judgment.

Meanwhile, we observe that the colluding behavior sharply affects the limited-copy routing protocol. In Figure 5.13, under Spray and Wait, the detection rate decreases more obviously than the detection rate using the unlimited-copy routing protocols. This is due to the fact that when fewer messages and ERs exist in the system, the bogus ERs have larger impact. Besides, when more colluding attackers join in the system, the detection rates also follow to decrease.

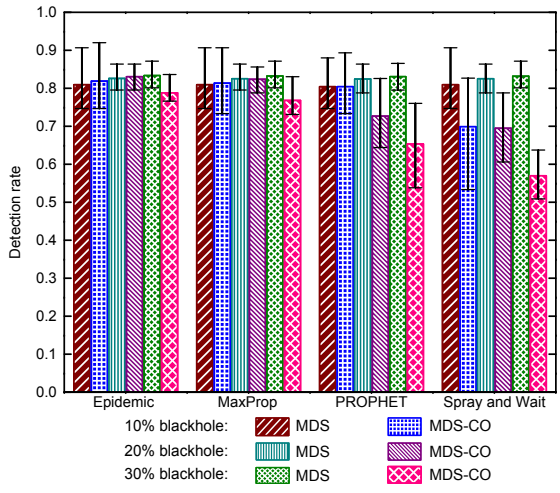
Comparing the adaptive threshold-based MDS with the cluster analysis based MDS, we find that under unlimited-copy routing protocols, the cluster analysis based MDS performs better especially when there is a larger number of attackers in the network. For limited-copy routing protocols, the cluster analysis based MDS obtains even higher detection rates compared to the adaptive threshold-based MDS (see Figure 4.6 (a)).



(a) The Helsinki scenario



(b) The San Francisco scenario



(c) The Braunschweig scenario

Figure 5.13: Detection Rate for Blackholes with and without Collusion – Using Cluster Analysis

In this section, we find that in some cases the colluding blackhole attackers can have destructive effect on the performance of the MDS. Although our MDS can efficiently detect malicious nodes in most cases, improving our MDS to cope with the colluding attackers is an important job to do in the future.

5.3.5 False Positive Rate

In this section, we will discuss about the problem of benign nodes mistakenly detected as evil nodes. The false positive rate is an important metric to denote how clear the MDS can discriminate between good and malicious nodes. Figure 5.14 and Figure 5.15 show the average and the maximum and minimum false positive rates versus the percentage of malicious nodes under four different routing protocols. In the Helsinki scenario, our MDS always achieves a zero false positive rate under all parameter combinations, showing the better results than the false positive rates using the adaptive threshold-based MDS (see section 4.3.3.3). Therefore, Figures 5.14 to 5.15 present the false positive rates for the San Francisco and Braunschweig scenario respectively.

In our system, when the drop probability of attackers is high, there is a distinctive difference between the behavior of normal and malicious nodes (see section 5.2.1.1). Hence, using cluster analysis our MDS can achieve a lower false positive rate under different routing protocols. For low drop probabilities, the behavior of malicious nodes is very similar to the behavior of normal nodes, therefore, it gets harder for cluster analysis to clearly discriminate between good and malicious nodes and thus the false positive rate is larger.

As shown in Figures 5.14 to 5.15, as the percentage of malicious nodes in the system increases, the average false positive rate will decrease. This is due to the fact that when normal nodes have more opportunities to meet malicious nodes, they can collect more data sample from the malicious nodes, helping them make a more precise decision.

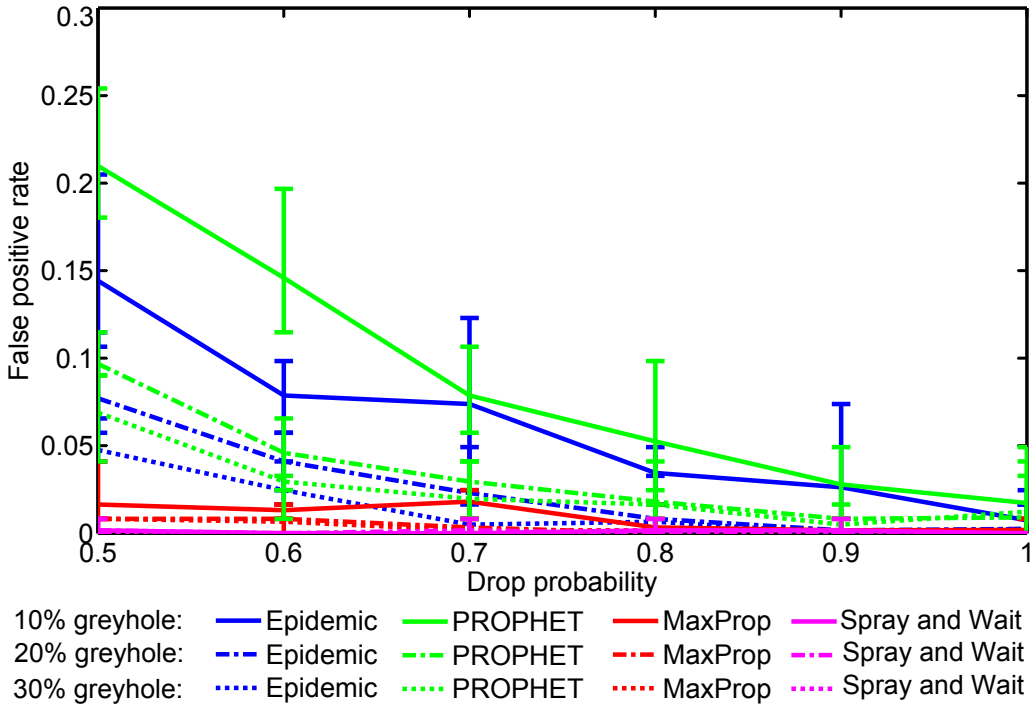
When we compare Figure 5.14 (a) with Figure 5.14 (b) and compare Figure 5.15 (a) with Figure 5.15 (b), we find that the FBL mechanism helps the system achieve a better false positive rate. First the FBL mechanism does not increase the false positive rate, as the required consensus among 3 friends before labeling a node as malicious prevents the propagation of false positives through the system. Besides, the FBL mechanism can help decrease the false positive rate since the assessment from 3 friends to label a node as friend can straighten out the misjudgement and make the node back to work in the system (see section 3.3.3.1).

In the San Francisco scenario, it can be seen that our MDS can achieve a low false positive rate using different routing protocols (see Figure 5.14 (a)): When the

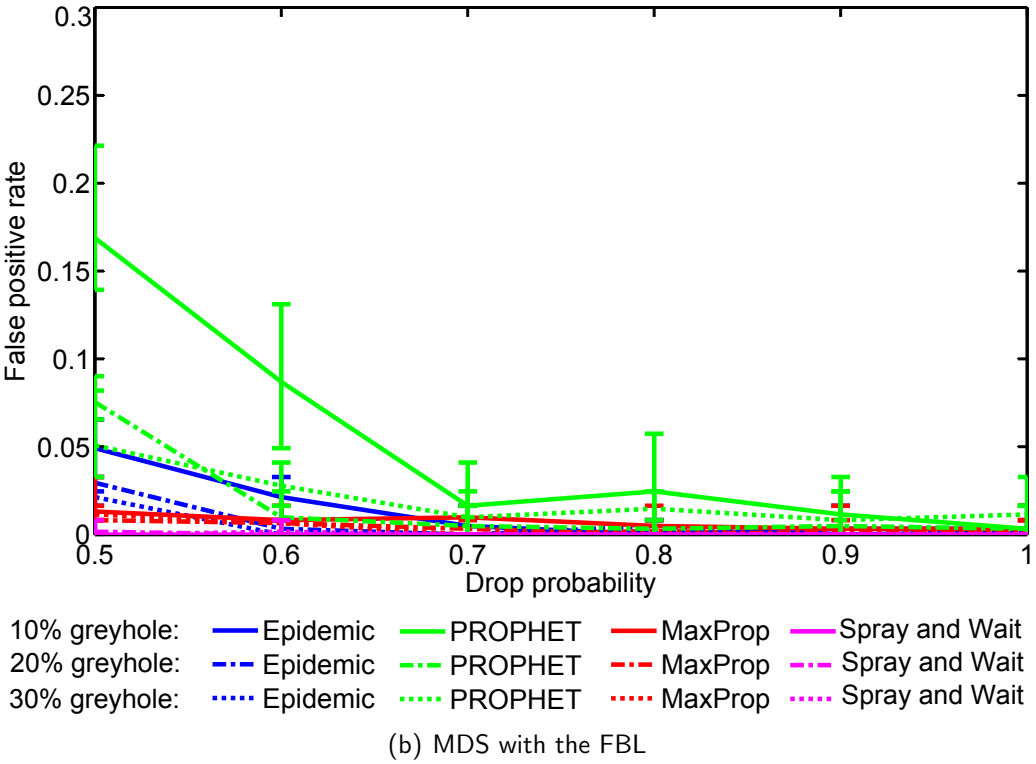
percentage of blackhole attackers ($p = 1$) varies from 10% to 30%, the average false positive rates change in the range of 0.25% to 0.74%, 0.09% to 0.18% and 1.23% to 1.72% under the routing protocols Epidemic, MaxProp and PROPHET respectively. For greyhole attackers, when the drop probability exceeds 0.7, the average false positive rate of our system is varying from 0 to 7.38%, 0 to 1.80% and 0.49% to 7.87% under the same routing protocols. As the greyhole attackers adopt lower drop probability ($p < 0.7$), the greyhole attackers have more similar behavior to the normal nodes, hence, the false positive rate increases a bit. Coping with blackhole attackers, our system achieves zero false positive under Spray and Wait. Meanwhile, when the greyhole attackers disturb the system, the average false positive rate of our system is less than 0.16% in different scenarios using Spray and Wait.

In Figure 5.14 (b), it shows that the MDS with the FBL can achieve a lower false positive rate using different routing protocols in the San Francisco scenario. When blackhole attackers exist in the network, the MDS with the FBL can achieve a false positive rate less than 0.33%, 0.33% and 0.33% using Epidemic, MaxProp and PROPHET respectively. When there are 10% greyhole attackers with a drop probability of 0.7, the false positive rate of the MDS with the FBL is 0.49%, 0.98% and 1.64%, reducing the false positive rates of the MDS by 93%, 45% and 79% for Epidemic, MaxProp and PROPHET. Comparing the MDS with the MDS with the FBL, the FBL can help friend nodes make a final decision considering friends' suggestion, avoiding the false judgement about suspected nodes. Using the FBL, the average false positive rate of our system is less than 4.92%, 1.31% and 16.89% for the same routing protocols in different scenarios. When the greyhole attackers have a high drop probability ($p \geq 0.7$), the MDS with the FBL obtains zero false positive under Spray and Wait.

We use Figure 5.15 to present the false positive rates in the Braunschweig scenario. It can be seen, that our MDS can achieve a low false positive rate using different routing protocols in Figure 5.15 (a): With blackhole attackers in the network, our system achieves zero false positive under different routings. Detecting distinctive greyhole attackers with higher drop probabilities, our system achieves a lower false positive rate. With drop probabilities decreasing, the false positive rates follow to increase. However, no matter that we vary the different number of attackers or we adopt different attack intensity, our MDS can achieve the average false positive rate less than 1.78%.

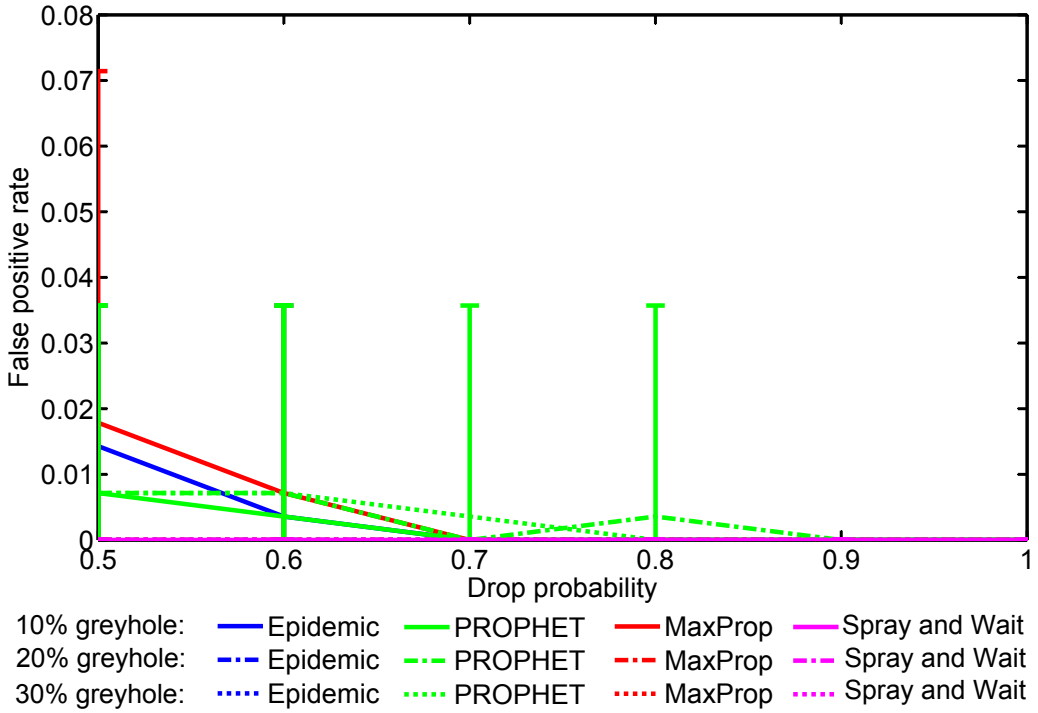


(a) MDS

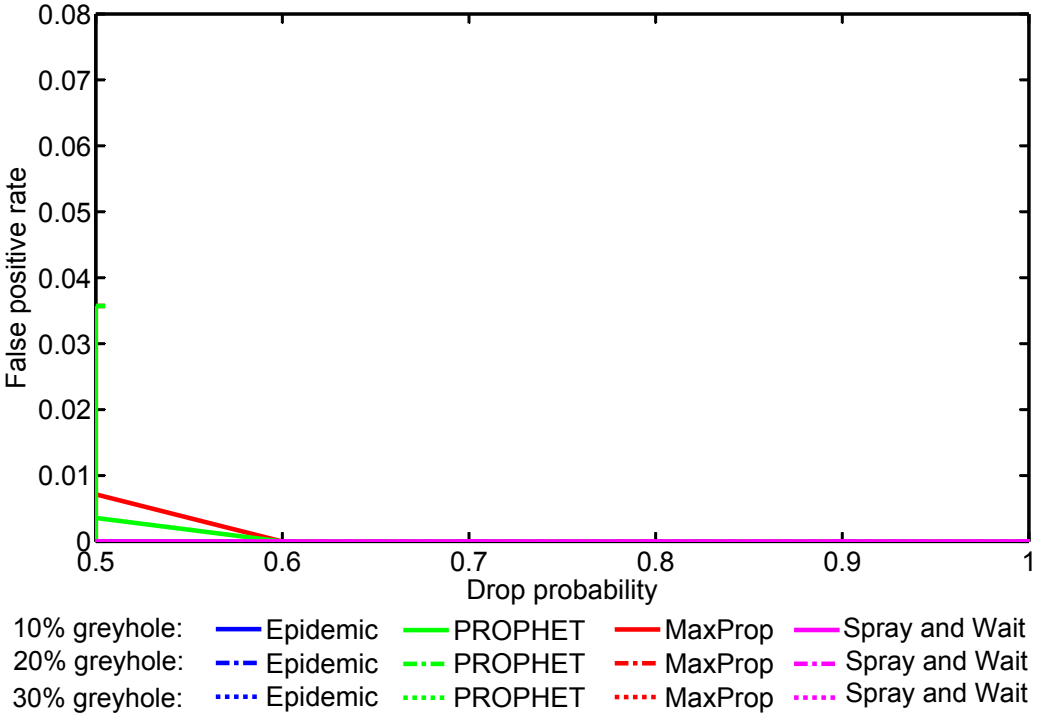


(b) MDS with the FBL

Figure 5.14: **False Positive Rate – Using Cluster Analysis in the San Francisco Scenario**



(a) MDS



(b) MDS with the FBL

Figure 5.15: **False Positive Rate – Using Cluster Analysis in the Braunschweig Scenario**

Using the FBL, Figure 5.15 (b) shows that our MDS can achieve zero false positive rate under different routings when the drop probability of attackers is greater than 0.7. Even if the attackers adopt lower drop probabilities, the false positive rate is less than 0.71% in different scenarios. The results show that in the Braunschweig scenario, the FBL mechanism also improves the performance of the MDS.

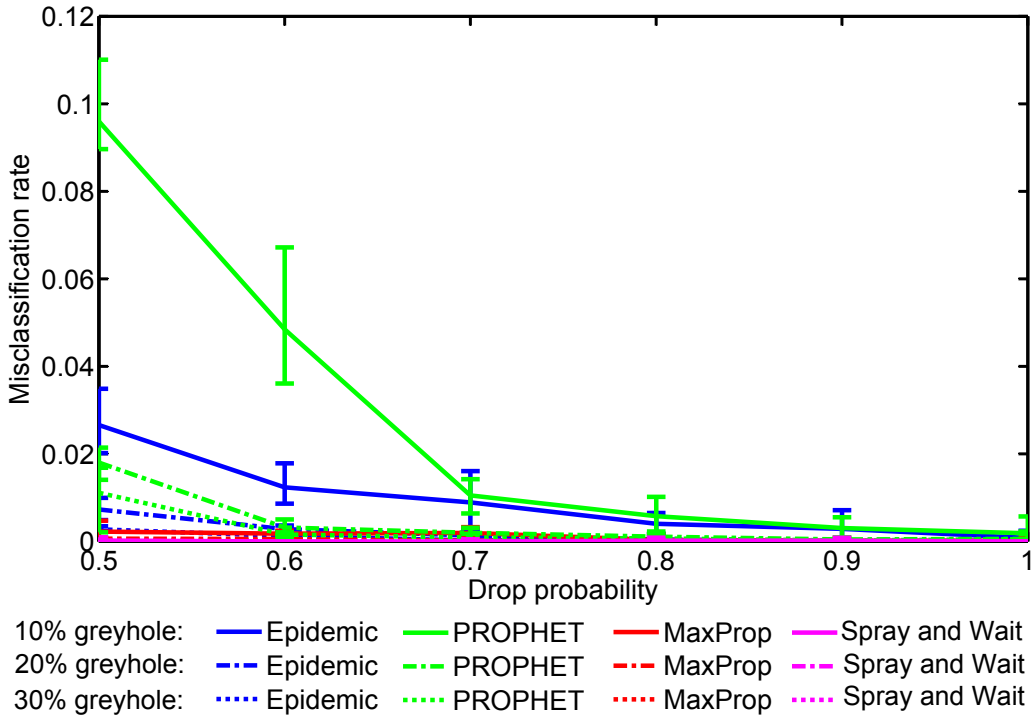
Comparing the detection rates in Figure 5.8 with the false positive rates in Figure 5.14 (a) and the detection rates in Figure 5.9 with the false positive rates in Figure 5.15 (a), we find that when the MDS achieves a higher detection rate, it also faces the problem with a higher false positive rate. For example, when drop probabilities are low, Epidemic, MaxProp and PROPHET keep high detection rates but meanwhile obtain high false positive rates. For Spray and Wait, although its detection rate is not so high, it can keep a low false positive rate compared to other routings.

In general, for classification problems there is a trade-off between detection rates and false positive rates. Considering the requirement in the realistic VDTNs, when we design the MDS, the classification parameters can be used to fine-tune the balance between desired detection efficiency and acceptable false positive rate.

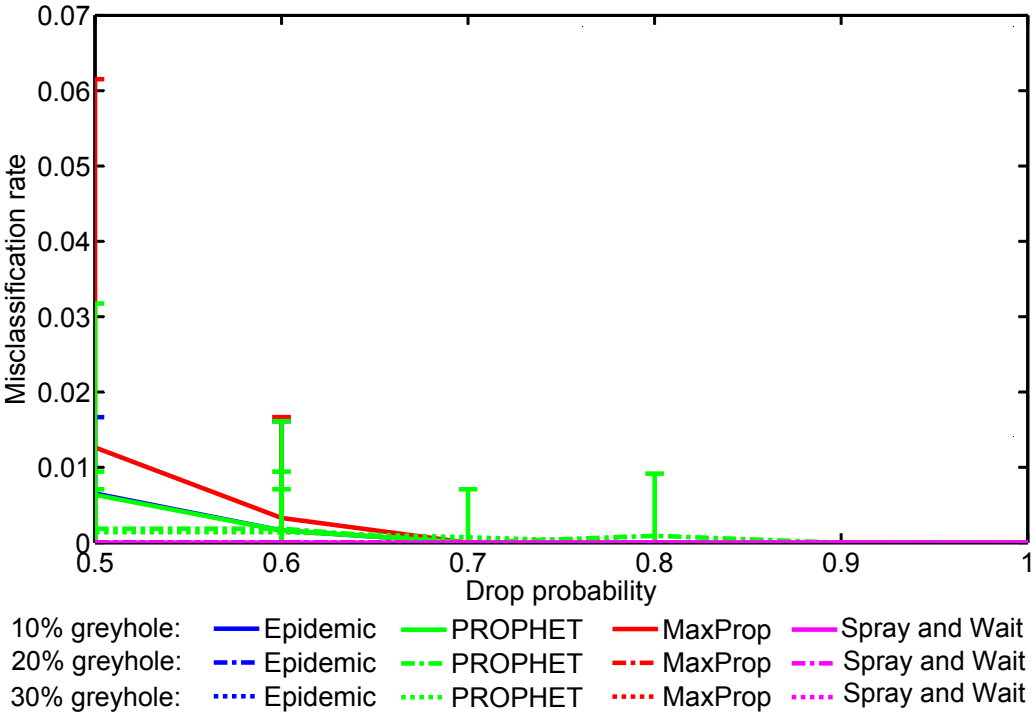
Our extensive simulations under different routing protocols demonstrate that our MDS can achieve a high detection rate (see section 5.3.2) and a low false positive rate for different scenarios.

5.3.6 Misclassification Rate

In section 5.3.5, according to our definition of the false positive rate, if a normal node is mistakenly detected by some or only one other node, this can increase the false positive rate. In a real scenario this means the mistakenly detected node loses the chance of communication with one other node for some time, but by no means it is excluded completely from the network. Hence, in Figure 5.16 we apply another metric to check the performance of our MDS. The misclassification rate shows for all individual positive (malicious) classifications of all nodes, how often the detection is wrong (for the definition see section 3.4.2).



(a) The San Francisco scenario



(b) The Braunschweig scenario

Figure 5.16: **Misclassification Rate – MDS Using Cluster Analysis**

Figure 5.16 (a) shows the misclassification rates in the San Francisco scenario. In this part, we will make the comparison between the false positive rates and the misclassification rates. In section 5.3.5, the results in Figure 5.14 (a) show that, when the system varies the percentage of the malicious nodes and the drop probabilities of the malicious nodes, MaxProp and Spray and Wait keep the false positive rates lower than 1.80% and 0.16% in all conditions. As shown in Figure 5.16 (a), in the San Francisco MaxProp and Spray and Wait scenarios the misclassification rates are smaller than 0.22% and 0.02% respectively under all conditions. In Figure 5.14 (a) we find higher false positive rates when attackers use low drop probabilities especially when using Epidemic and PROPHET in the San Francisco scenario. For Epidemic, when there are 10% greyhole attackers with the drop probability equal to 0.5, although the result in Figure 5.14 (a) shows a false positive rate of 14.43%, actually there are less than 2.66% misclassifications. Using PROPHET, detecting 30% greyhole attackers with the drop probability of 0.5, the system obtains the false positive rate of 6.89% but the misclassification rate is 1.11%. The results in different conditions show that most of the time a node will be correctly classified. The remaining misclassifications might slightly impact the performance of the network locally, but will not hurt the system globally.

In Figure 5.16 (b) we apply the misclassification rate to check the performance of our MDS in the Braunschweig scenario. As shown in Figure 5.16 (b), our MDS maintains a low misclassification rate. Epidemic, MaxProp and PROPHET can keep the average misclassification rate smaller than 0.65%, 1.26% and 0.63% respectively. Spray and Wait obtains zero misclassification rate in all condition. For PROPHET, when there are 30% greyhole attackers with a drop probability equal to 0.5, although Figure 5.15 (b) in section 5.3.5 shows a false positive rate of 0.71%, actually there are less than 0.14% misclassifications. These low average misclassification rates denote that most of the time our MDS can correctly classify the nodes. In the Braunschweig scenario, the lower the misclassification is, the more slightly it can locally impact the performance of the network.

In section 5.3.5, we use the false positive rate to discuss about the percentage of benign nodes mistakenly detected as evil nodes. In this part, we apply the misclassification rate to denote the percentage of wrong detections in relation to the total number of detections. Both of the metrics show that our MDS can keep the low misjudgement about the benign nodes.

5.3.7 Detection Speed

The detection speed is an important metric to evaluate our MDS. Only after a malicious node has been detected, measures to mitigate its impact can be taken. Mean-

while, a fast detection speed shows the system can promptly eliminate the negative effects from attackers.

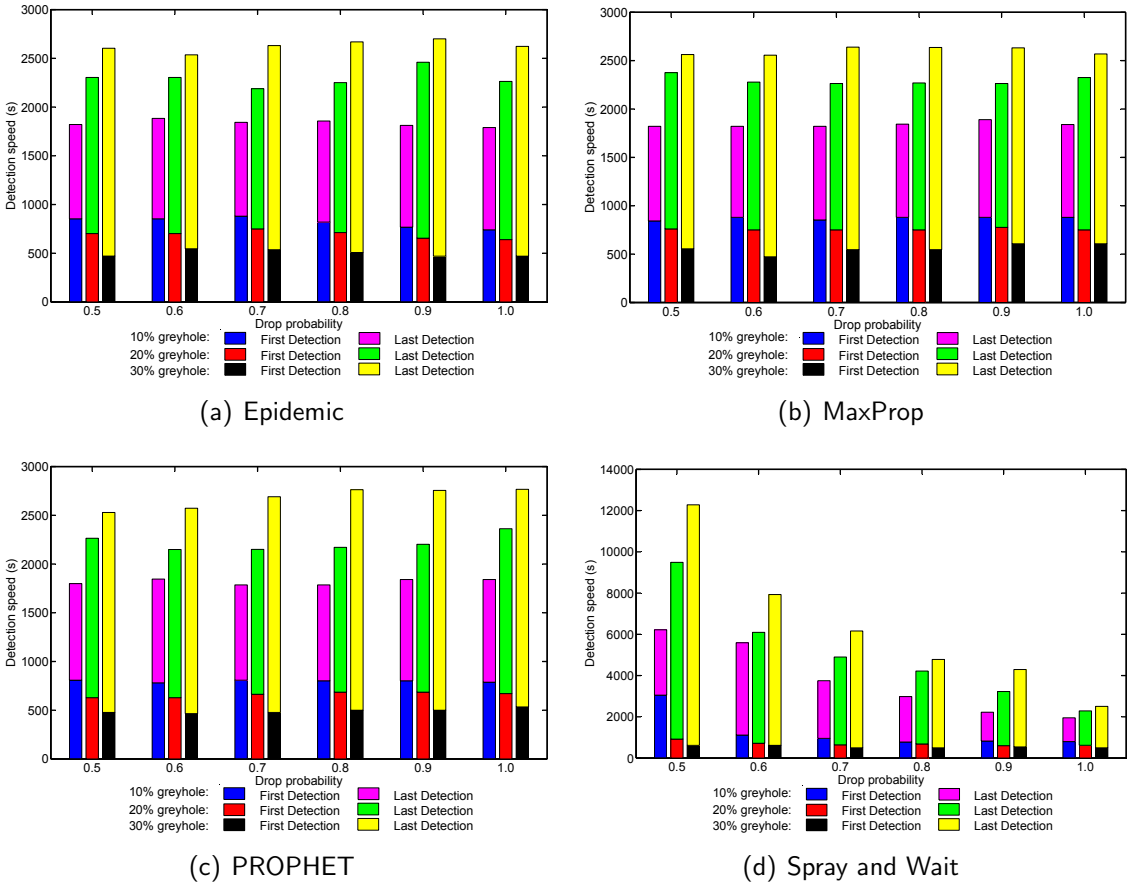
Figures 5.17 to 5.19 show the detection speed in the Helsinki, San Francisco and Braunschweig scenario respectively. We vary the drop probabilities as well as the routing protocols to check whether our MDS has a fast detection speed. As introduced in section 4.3.3.4, "First Detection" is defined as the time it takes for the first malicious node to be detected by a normal node. "Last Detection" is defined as the time after which all of the malicious nodes have been detected at least once by a normal node. This time is also the lowest time boundary for achieving 100% detection rate, assuming the system could support perfect knowledge sharing among all nodes. For this simulation it makes no difference whether the system allows friend nodes to exchange the LBL information, as a node needs to be detected by at least one node before it has the chance to add to another node's FBL. Hence, Figures 5.17 to 5.19 only show the detection speed of our MDS without using the FBL principle (see section 3.3.3.1).

As seen in Figure 5.17, 5.18 and 5.19, there are several empirical regularities we find out in our detection speed: Firstly our results show that the detection speed of blackhole attackers is faster than the speed of greyhole attackers since blackhole attackers have more obviously malicious behavior compared to greyhole attackers; when the drop probability varies from 1.0 to 0.5, the total detection speed takes longer. This is due to the fact that facing the malicious nodes with lower drop probabilities, the MDS needs to wait longer until enough information can be accumulated in form of the ERs, and then makes a correct judgement of the nodes; when the percentage of malicious nodes varies from 10% to 30% under the same drop probability, we find the "First Detection" becomes much faster. This is because when our MDS obtains more information from the malicious node group, the K-means clustering can much more accurate and faster cluster the behavior of malicious nodes. Hence, the "First Detection" will be faster compared to the condition that there are less malicious nodes in the system; in addition, detecting a higher number of attackers generally takes a longer time. Hence, when the percentage of malicious nodes varies from 10% to 30% under the same drop probability, the total detection speed increases.

Figure 5.17 shows the detection speed of our MDS when the attackers using different drop probabilities exist in the Helsinki scenario. It can be seen that when there are 10% blackhole attackers ($p = 1$) in the system, the MDS takes about 741 s, 880 s, 788 s and 800 s to detect the first malicious node and about 1790 s, 1839 s, 1841 s and 1952 s to detect each of the malicious nodes under the routing protocols Epidemic, MaxProp, PROPHET and Spray and Wait respectively. As the density of blackhole attackers increases from 10% to 30%, the MDS needs more time to detect all of the different malicious nodes. When the greyhole attackers ($0 < p < 1$) adopt different

drop probabilities, our MDS can keep a stable detection speed in Epidemic, MaxProp and PROPHET. Even when there are 30% greyhole attackers using different drop probabilities in the system, the total detection speed does not exceed 2700 s, 2640 s and 2767 s using Epidemic, MaxProp and PROPHET respectively. For routing protocols that exchange and generate limited messages such as Spray and Wait, the detection speed becomes slow especially when greyhole attackers adopt lower drop probabilities. The reason is that the limited messages in the system lead to the limited information accumulated in the ERs, increasing the difficulty for nodes to make the judgement, hence, the detection time becomes longer in Spray and Wait.

Compared to the detection speed using the adaptive threshold-based MDS in section 4.3.3.4, our MDS using the cluster analysis improves the detection speed no matter varying the number of attackers, the attack intensity or the employed routing protocols.



(a) Epidemic (b) MaxProp

(c) PROPHET (d) Spray and Wait

Figure 5.17: Detection Speed – Using Cluster Analysis in the Helsinki Scenario

The detection speed of the San Francisco scenario is shown in Figure 5.18. The first detection speed of the San Francisco scenario is 1105 s, 1382 s, 1307 s and 2220 s under four different routing protocols Epidemic, MaxProp, PROPHET and Spray and Wait respectively with 10% blackhole attackers in the system. And it takes around 15023 s, 14649 s, 16671 s and 18940 s to detect all blackhole attackers under the same condition. When there are 30% greyhole attackers ($0 < p < 1$) in the system, our MDS can keep a stable detection speed and the total detection speed is under 25535 s, 25899 s and 27008 s using Epidemic, MaxProp and PROPHET. Spray and Wait has the longest detection time in the San Francisco scenario too, especially when the greyhole attackers use lower drop probabilities. This is due to the fact that in contrast to the other protocols, Spray and Wait strictly limits the number of copies per data item. With limited messages in the system, the MDS needs to wait longer until enough information has been accumulated in form of the ERs and then makes the decision.

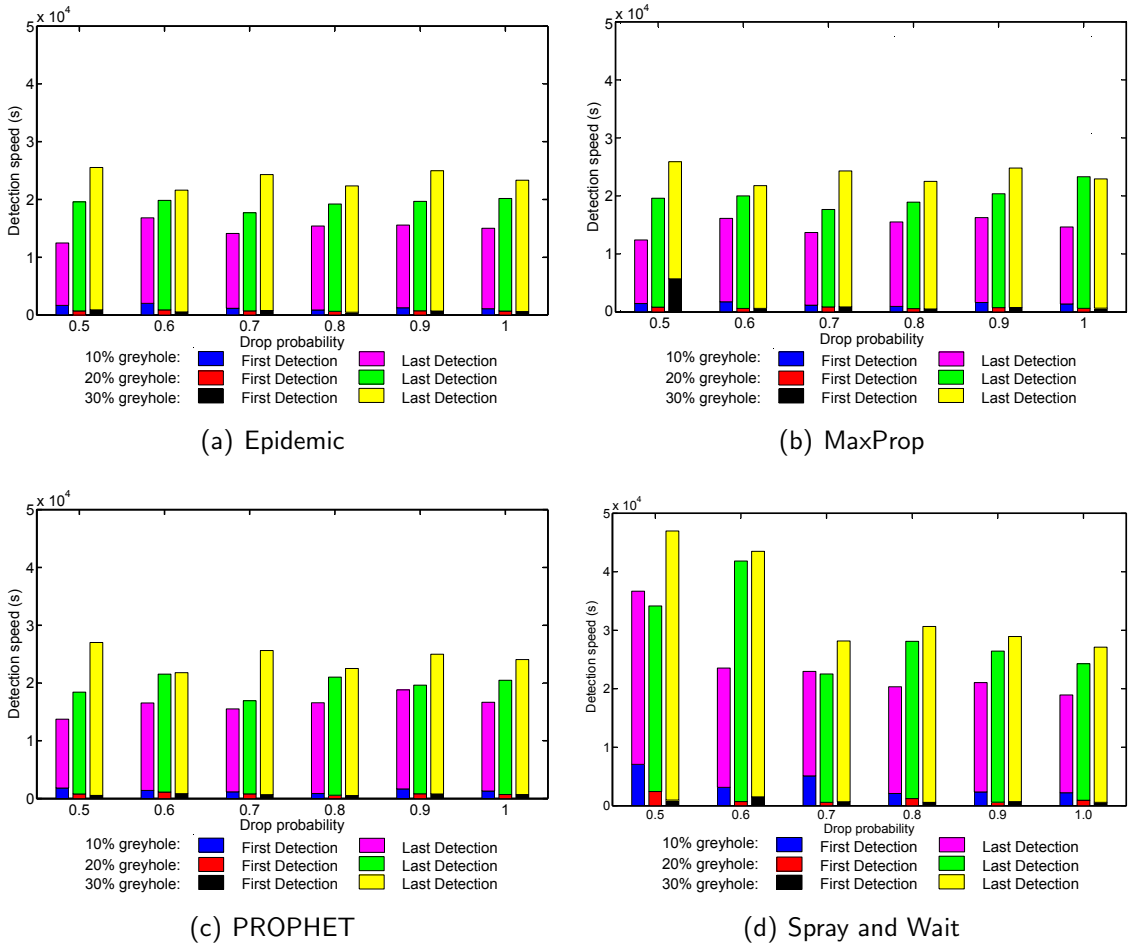


Figure 5.18: **Detection Speed – Using Cluster Analysis in the San Francisco Scenario**

Comparing Figure 5.17 with Figure 5.18, it can be seen that in the San Francisco scenario, the first detection speed and the total detection speed are not as fast as the results in the Helsinki scenario. This is due to the fact that the San Francisco scenario is a much sparser environment with larger area and larger number of nodes. Vehicular nodes need to take some time to encounter other nodes and then can judge other nodes' behavior.

In the Helsinki and San Francisco scenario, the vehicular nodes generate and transmit messages between each other. While in the Braunschweig scenario, the vehicular nodes transfer all messages to the gateway. Although our MDS applies in different scenarios, it works effectively. As seen from Figure 5.19, the detection speed of blackhole attackers is faster than the speed of greyhole attackers. When there are 10% blackhole attackers ($p = 1$) in the system, our MDS takes 3680 s, 3680 s, 3994 s and 3756 s to detect each of the blackhole attackers under four different routing protocols. Besides, when detecting greyhole attackers our MDS can keep a stable detection speed in Epidemic, MaxProp and PROPHET. Spray and Wait has the longest detection time as in the Helsinki and San Francisco scenario.

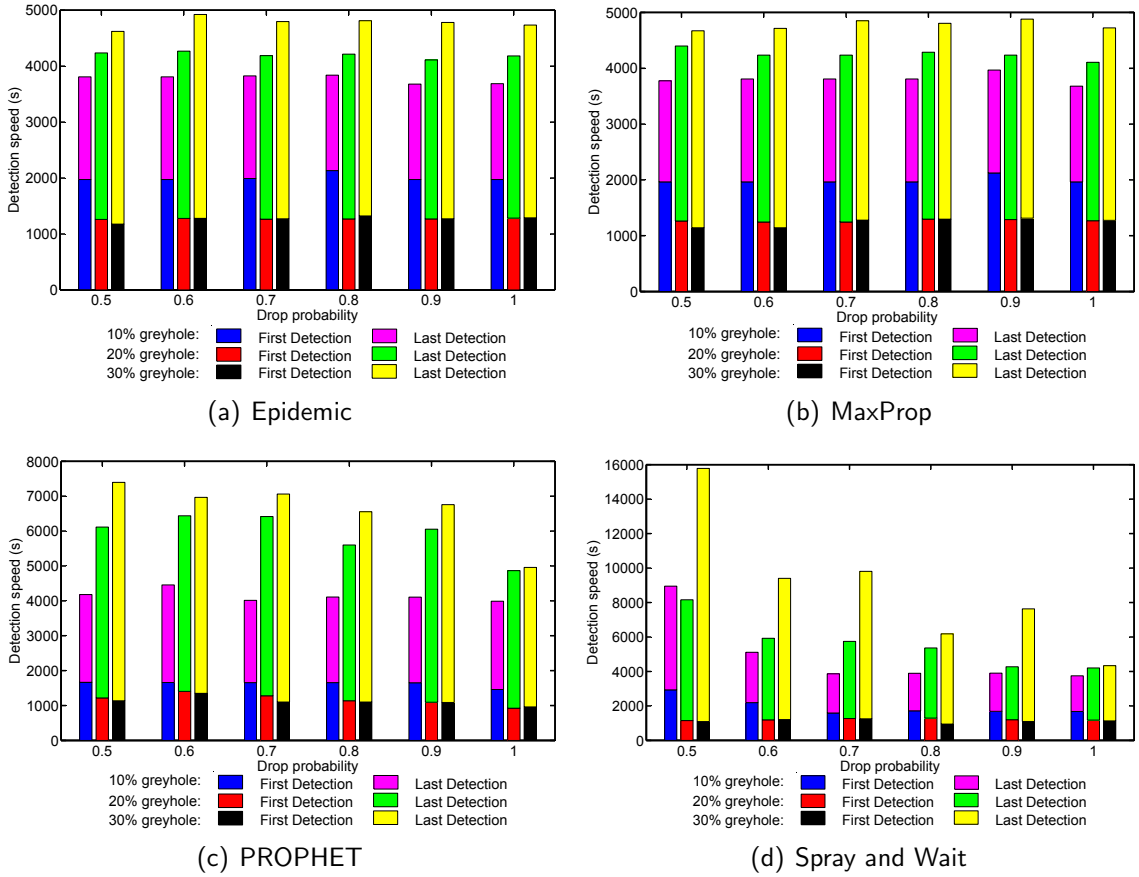


Figure 5.19: **Detection Speed – Using Cluster Analysis in the Braunschweig Scenario**

As seen from Figures 5.17 to 5.19, our simulation results show that our proposed MDS has a well reaction to blackhole and greyhole attackers. With a good detection speed, our MDS tries to quickly exclude malicious nodes from participating in the network. In general, when the size of the scenario is different or the scenario is composed of different number of vehicular nodes, the first detection speed and the total detection speed will vary a bit according to the characteristics of scenarios.

5.3.8 Relayed Messages

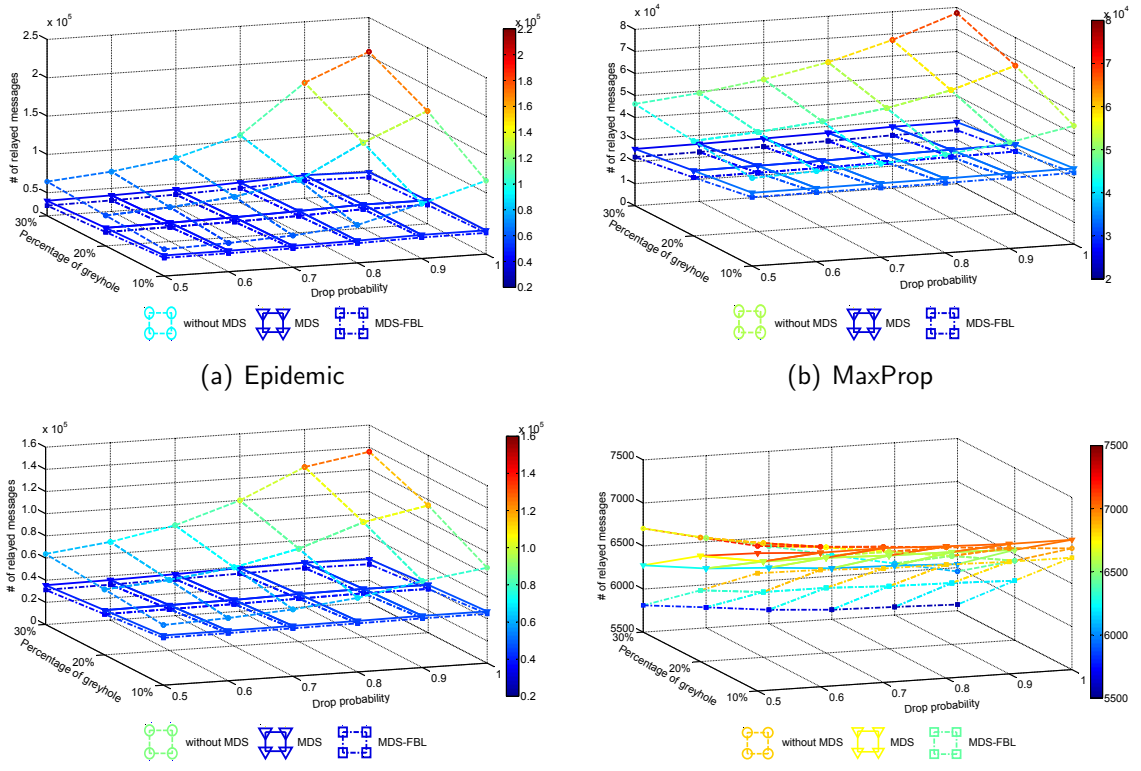
The number of relayed messages has a direct relation with the energy consumption of a network, as each message needs a certain amount of energy to be sent, received and processed. While energy consumption might be a secondary concern in VDTNs, more importantly less transmitted messages also mean a less congested MAC layer. Therefore, reducing useless transmissions directly leads to a better quality of service and more capacity for applications.

Figures 5.20 to 5.22 present the average number of messages relayed in the Helsinki, San Francisco and Braunschweig scenario versus the percentage of malicious nodes using different drop probabilities in the system under four different routing protocols. The x-axis shows the drop probability. The y-axis shows the percentage of evil nodes in the system. The z-axis shows the total number of relayed messages in the system. Because the number of relayed messages under four routing protocols and three scenarios shows significant difference with each other, hence, different coordinate intervals for z-axis are chosen for each of the figures.

In these three scenarios, Figures 5.20 (a) to 5.20 (c), Figures 5.21 (a) to 5.21 (c) and Figures 5.22 (a) to 5.22 (c) show that, especially for routing protocols with unlimited replication such as Epidemic, MaxProp or PROPHET, blackhole or greyhole attackers can cause a drastic increase in relayed messages. When the percentage of malicious nodes varies from 10% to 30% under the same drop probability, the total number of relayed messages will increase in the system. Using Epidemic, the nodes in the system will try to transfer their messages to any nodes which still do not store these messages, making the system keep the highest number of relayed messages without using our MDS in these three scenarios. In MaxProp and PROPHET, the nodes will transfer the messages to the nodes which are chosen according to their routing mechanism rather than transfer the messages to any random node. Hence, when blackhole or greyhole attackers exist in these systems, blackhole or greyhole attackers cannot affect the number of relayed messages as drastic as it is the case for Epidemic. Whether varying the drop probability or the percentage of evil nodes in the system, our MDS always achieves a lower number of relayed messages compared to the number of relayed messages without using the MDS. Adding the FBL mechanism decreases

the wasted messages sent to attackers hence leverages even more relayed message savings.

Routing protocols which limit the number of replicas such as Spray and Wait do not suffer so much from increased relaying, see Figures 5.20 (d), 5.21 (d) and 5.22 (d). In our setup each message is allowed to be copied 6 times by Spray and Wait, hence, the total number of relayed messages has an upper bound. When the malicious nodes drop messages, the number of relayed messages will be decreased. As our MDS or the MDS with the FBL decreases the chance that the limited replicas are relayed to the evil nodes, the total number of relayed messages by using our MDS or the MDS with the FBL is close to the upper bound. For routing protocols with a limited number of replicas this is the desired behavior, as it increases the probability that messages can be forwarded towards their destinations and thus increases the delivery rate in the system.



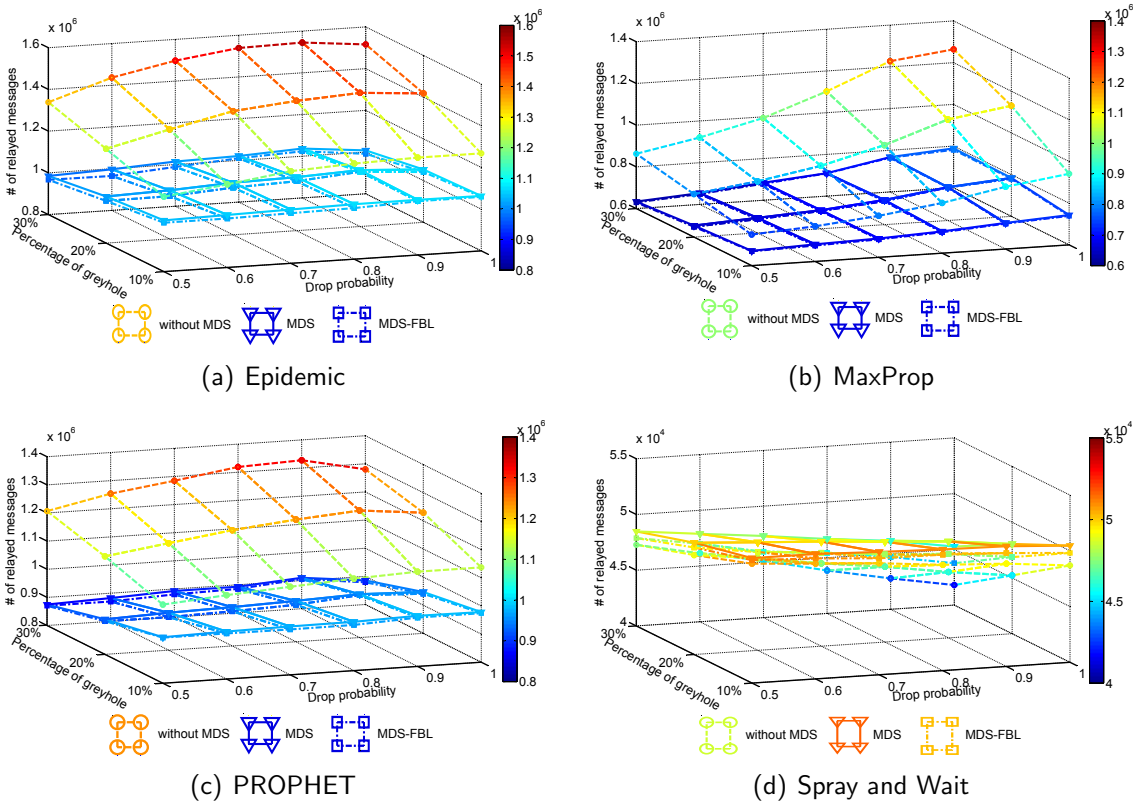
(a) Epidemic (b) MaxProp

(c) PROPHET (d) Spray and Wait

Figure 5.20: **Relayed Messages – Using Cluster Analysis in the Helsinki Scenario**

For unlimited-copy routing protocols, in the case there are 30% blackhole attackers in the system, compared to the condition without MDS, our MDS reduces relayed messages by 76.5%, 63.9% and 71.3% for Epidemic, MaxProp and PROPHET respectively in the Helsinki scenario, 33.5%, 37.6% and 30.8% for the same routing

protocols in the San Francisco scenario and 62.2%, 58.6% and 56.1% for the same routing protocols in the Braunschweig scenario. In the Helsinki scenario, these results show that our cluster analysis based MDS performs better than the adaptive threshold-based MDS with the relayed message savings of 64.0%, 54.1% and 59.6% in Figure 4.9. Using the FBL mechanism, coping with 30% of blackhole attackers in the system, our MDS with the FBL obtains better results. The MDS with the FBL reduces relayed messages by 79.8%, 68.6%, 74.6% for Epidemic, MaxProp and PROPHET respectively in the Helsinki scenario, 34.6%, 38.1% and 31.4% in the San Francisco scenario and 64.5%, 61.1% and 57.7% in the Braunschweig scenario.



(a) Epidemic (b) MaxProp

(c) PROPHET (d) Spray and Wait

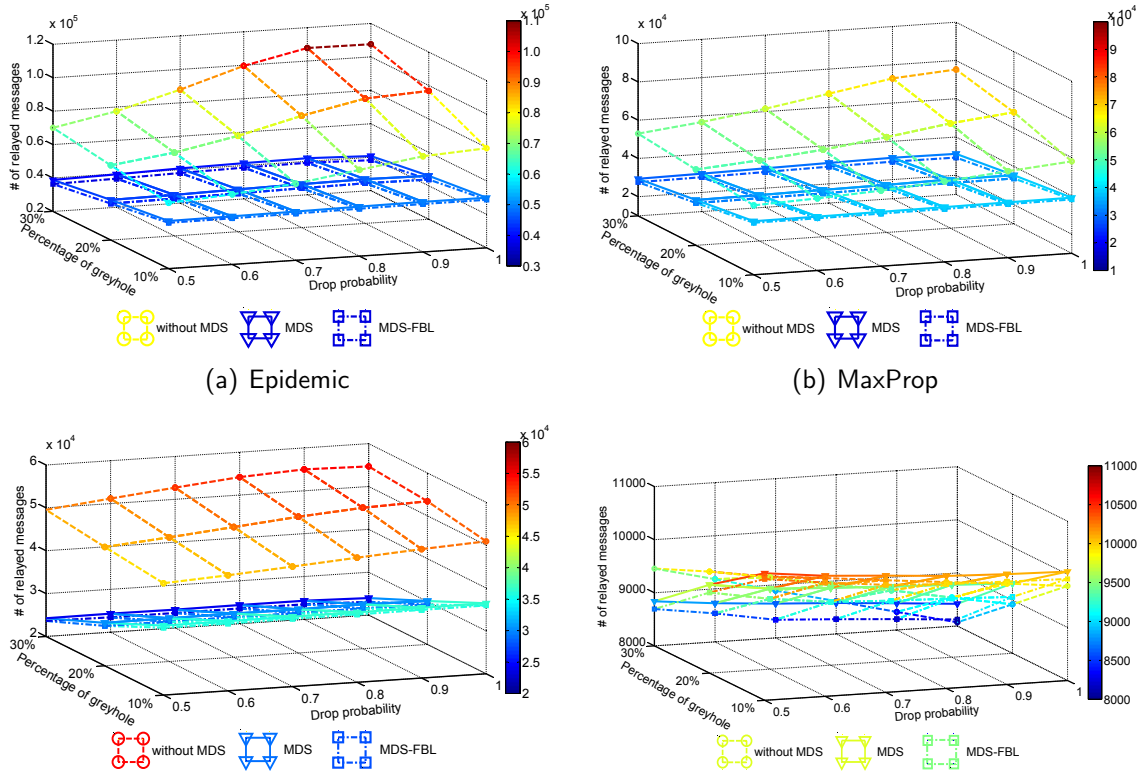
Figure 5.21: **Relayed Messages – Using Cluster Analysis in the San Francisco Scenario**

When attackers use lower drop probabilities, their abilities to destroy the network also decrease, hence, the number of relayed messages follows to decrease. When there are 30% greyhole attackers with the drop probability of 0.5 in the system, compared to the relayed message results without MDS, our MDS can reduce 38.8%, 44.2% and 44.3% relayed messages for Epidemic, MaxProp and PROPHET respectively in the Helsinki scenario, 26.4%, 26.5% and 27.5% in the San Francisco scenario and 44.5%, 44.5% and 51.1% in the Braunschweig scenario. The MDS with the FBL reduces more relayed messages, 48.6%, 52.2% and 50.9% in the Helsinki scenario, 27.8%,

27.0% and 27.9% in the San Francisco scenario and 48.0%, 47.9% and 52.3% in the Braunschweig scenario.

Although in the San Francisco scenario the relative decrease of relayed messages is smaller than in the Helsinki scenario or in the Braunschweig scenario, in fact, the total number of relayed message in the San Francisco scenario is much larger.

Finally, our MDS or the MDS with the FBL can significantly reduce the number of relayed messages for common DTN routing protocols, which create a large number of replicas. Meanwhile, our MDS or the MDS with the FBL can protect the limited replicas from transmitting to malicious nodes under limited-copy routing protocols. By reducing a large number of relayed messages, our MDS or the MDS with the FBL saves a lot of energy and precious MAC layer resources.



(a) Epidemic (b) MaxProp

(c) PROPHET (d) Spray and Wait

Figure 5.22: Relayed Messages – Using Cluster Analysis in the Braunschweig Scenario

5.4 Performance Evaluations in Heterogeneous VDTNs

In section 5.3, we use the clustering algorithm and subsequent rules to identify the malicious nodes. These methods have proven to adapt successfully to different vehicular scenarios, with homogenous nodes.

Today, as smartphones are becoming computationally more powerful and offer a variety of communication interfaces, it becomes attractive to investigate whether smartphones and vehicular nodes can cooperate with each other, forming a network that can provide better quality of service to applications. Motivated by the EMMA project (see section 2.1), in this section we propose a hybrid urban communication network consisting of buses, trams, stationary gateways and pedestrians.

Buses and trams move within the whole metropolitan area measuring air pollution and sending the collected data to the gateway. We assume the DTN hardware (see Figure 5.23) installed in buses and trams is similar to the equipment used in the project EMMA (see section 2.1). The current generation of these nodes are equipped with a GPS receiver, a power supply, a specialized environmental sensor and a DTN board running embedded Linux (OpenWRT) with IEEE 802.11 (WiFi) transceivers. The sensor is responsible for measuring air pollution data, the DTN board combines this data with GPS positioning information and a timestamp and sends this data using the IBR-DTN Bundle Protocol implementation [39].

The gateway is a stationary node which is responsible for gathering the air pollution data and sending it to the data center for further analysis. Pedestrians with smartphones walk around in the city. With such a system it would be easy to integrate pedestrians using smartphones as they could run the Android version of IBR-DTN [113] to communicate with the vehicular nodes. Pedestrians have the ability to generate their own messages and forward other buses', trams' or pedestrians' messages.

In this section, we evaluate the feasibility of creating an integrated DTN consisting of smartphones and the public transportation system. Most importantly the question we are interested in this work is, whether our MDS using the clustering algorithm is still suitable when there are nodes with widely varying characteristics such as the movement or communication patterns within a single system. The evaluation results will show whether our MDS is able to efficiently detect attackers and defend the hybrid network against the interference of malicious nodes.

Figure 5.23: The DTN Hardware for a Tram

5.4.1 Simulation Setup

We still use the DTN simulator *The ONE* to evaluate the system. The simulated time is 12 hours (43 200 s). Each node has 1 GB buffer size, a transmission radius of 50 meters and a transmission rate of 250 kB/s. Each message has a lifetime of 45 minutes and a size between 500 kB and 1 MB. We performed comparative measurements using the routing protocols Epidemic, MaxProp, PROPHET and Spray and Wait. Unless otherwise noted, the simulation results presented for each scenario are the average, min and max results of 10 experimental runs.

We simulate the Braunschweig public transportation system. The system consists of 54 stops and 28 buses and trams running on 13 assigned lines. Vehicles move with a speed between 9.72 km/h and 50.04 km/h. In an interval between 25 and 30 seconds, a randomly chosen vehicle generates one air pollution message which is sent to the gateway. The gateway, located near the Braunschweig main station, receives those messages. We add 280 pedestrians with a moving speed varying from 1.04 m/s to 1.51 m/s using the Random Waypoint movement model. In an interval between 15 and 25 seconds, a randomly chosen pedestrian generates one message for another pedestrian (see Table 5.3).

In this hybrid VDTN, our system will focus on detecting and mitigating blackhole attackers, which may either be the result of a deliberate attack or the result of hard- or software failures.

Table 5.3: **Simulation Parameters in the Hybrid Braunschweig Scenario**

Value	Hybrid Braunschweig Scenario
# vehicular nodes	28 vehicular nodes + 1 gateway
# pedestrians	280
Vehicular movement	simulated movements based on real world context data
Pedestrian movement	random waypoint movement model
Transmission radius	50 m
Buffer size	1 GB
Simulation time	12 h
Malicious Nodes	10%, 20%, 30%
Drop probability	1.0

5.4.2 Pedestrians Supporting the VDTN

As lined out in section 5.4.1, we consider an application where vehicles gather environmental data and send it to a gateway. In this experiment we will check the effect of adding pedestrians to the system and compare the performance to a system using vehicles alone.

5.4.2.1 Delivery Rate in the VDTN

The delivery rate is an indication of the service quality in VDTNs. Figure 5.24 presents the delivery rates for five different scenarios using four different routing protocols. In all five scenarios, only vehicles generate messages and send them to the gateway. In the first and fifth scenarios there are only vehicles in the system, and we evaluate the delivery rates with 100% benign vehicles and 20% malicious vehicles respectively. In the other scenarios, pedestrians are added to the system to help forwarding the messages. None, 50% or 100% malicious pedestrian nodes are chosen to evaluate the pedestrians' effect on network performance.

The results depicted in Figure 5.24 show that, in the first scenario when there are only vehicles in the network, they achieve delivery rates of 87%, 95%, 84% and 78% for Epidemic, MaxProp, PROPHET and Spray and Wait respectively. When we add 280 benign pedestrians to the network, the delivery rates for Epidemic and MaxProp increase. Even with 50% malicious pedestrian nodes, the system still sustains a higher delivery rate for Epidemic and MaxProp. In these cases, only when 100% of the pedestrian nodes are malicious nodes, the delivery rates decrease because these

malicious nodes replace some communicating opportunities between vehicles while dropping all received messages.

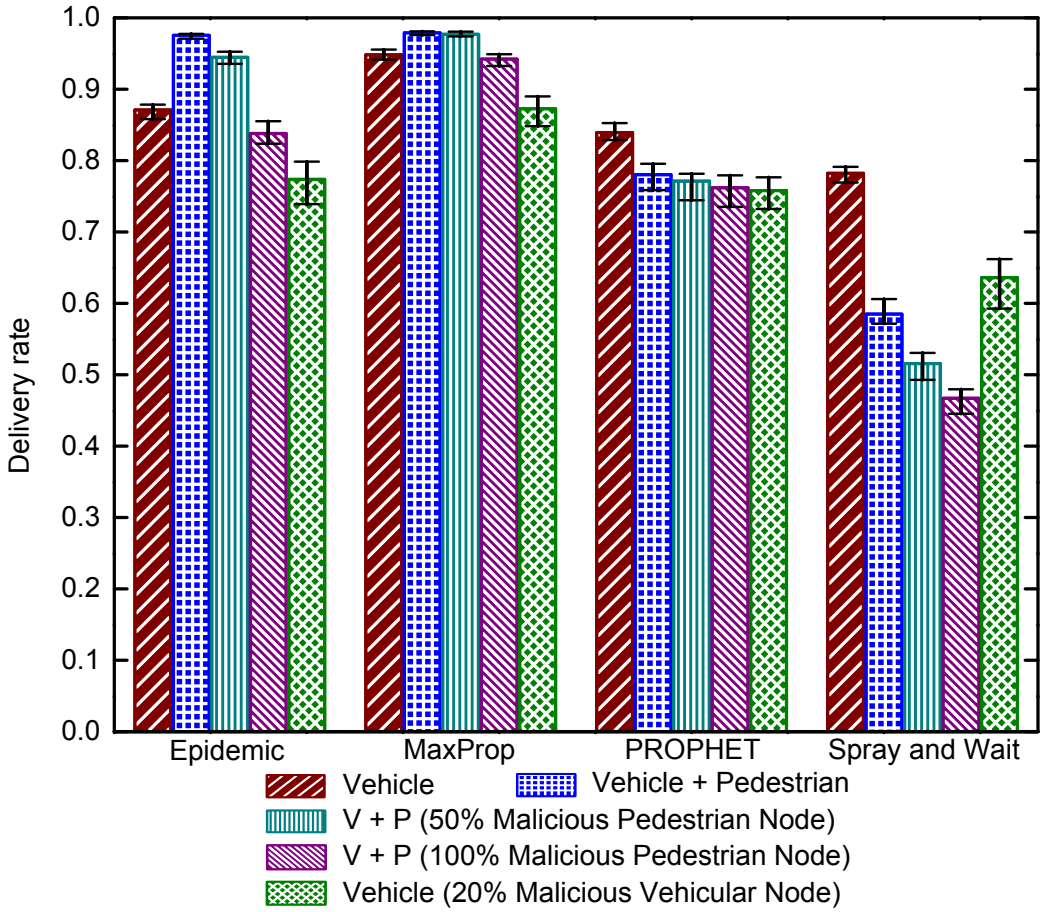


Figure 5.24: Delivery Rate in the VDTN

The results obtained from Epidemic and MaxProp show, that the network performance can be improved with the help of pedestrians. However, when using PROPHET or Spray and Wait, adding pedestrian nodes decreases the delivery rates. The reason for PROPHET's performance is that it is designed to cope with non-random mobility models. When there are only vehicles in the network, PROPHET can achieve a good delivery rate because of the repeating behavioral patterns of buses and trams. However, with randomly moving pedestrians, PROPHET has difficulties predicting good forwarding nodes, hence the delivery rates decrease.

The problem for Spray and Wait is, that this protocol limits the number of replicas for a given bundle. Giving one of the limited bundle copies to a vehicle is normally a good idea, as vehicles travel faster and by design the bus and tram lines usually cover a large area. However, if a pedestrian gets a copy, he moves much slower and

might not cover a large area due to the Random Waypoint movement model. Thus, there is a higher chance for a bundle to get "stuck".

We also checked the delivery rates when there are 20% malicious vehicular nodes in the network. This decreases delivery rates significantly in all cases. This shows the relative importance of the vehicles in the examined scenario.

Overall we see, that with a suitable routing protocol, pedestrians can be used to improve the performance of the system. Even a substantial amount of misbehavior in the pedestrian nodes will at least not hurt the performance in term of delivery rates. Misbehavior in the vehicles on the other hand is more critical, and should be detected reliably.

5.4.2.2 Latency in the VDTN

Another important performance metric is the latency (for the definition see section 3.4.2). Figure 5.25 presents the latency for the five different scenarios using four different routing protocols. We see that with Epidemic and MaxProp the 280 additional pedestrians decrease latency, even with 50% malicious pedestrian nodes. Using Epidemic and MaxProp, the latency with 280 benign pedestrians joining the network, is shorter than the latency when only vehicles exist in the network. Only with 100% malicious pedestrian nodes the latency increases a bit. As in the delivery rate experiments, 20% malicious vehicular nodes have a more severe effect and affect the latency negatively.

As seen in the delivery rate results, PROPHET does not cope very well with the random mobility model, hence, the latency when random pedestrian nodes join the network increases.

Interestingly, with pedestrians joining in the network, the latency time using Spray and Wait also decreases. However, this does not imply higher performance: Keep in mind, that with Spray and Wait the delivery rate decreased significantly with additional pedestrians. Spray and Wait [55] implements a mechanism that distributes a fixed, limited number of copies first and then waits for one of the relays encountering the destination. Since giving one of the bundles to a random moving pedestrian that might not cover much area is an inferior choice to giving that bundle to a vehicle exhibiting fast and directed motion, the delivery rate decreased. On the other hand, this implies that only the "easy" bundles get delivered in this scenario: Either the bundle is generated near the gateway from a network topology point of view, or it has a high chance of getting lost due to the limited lifetime. Therefore, the average latency for the bundles that actually do arrive decreases.

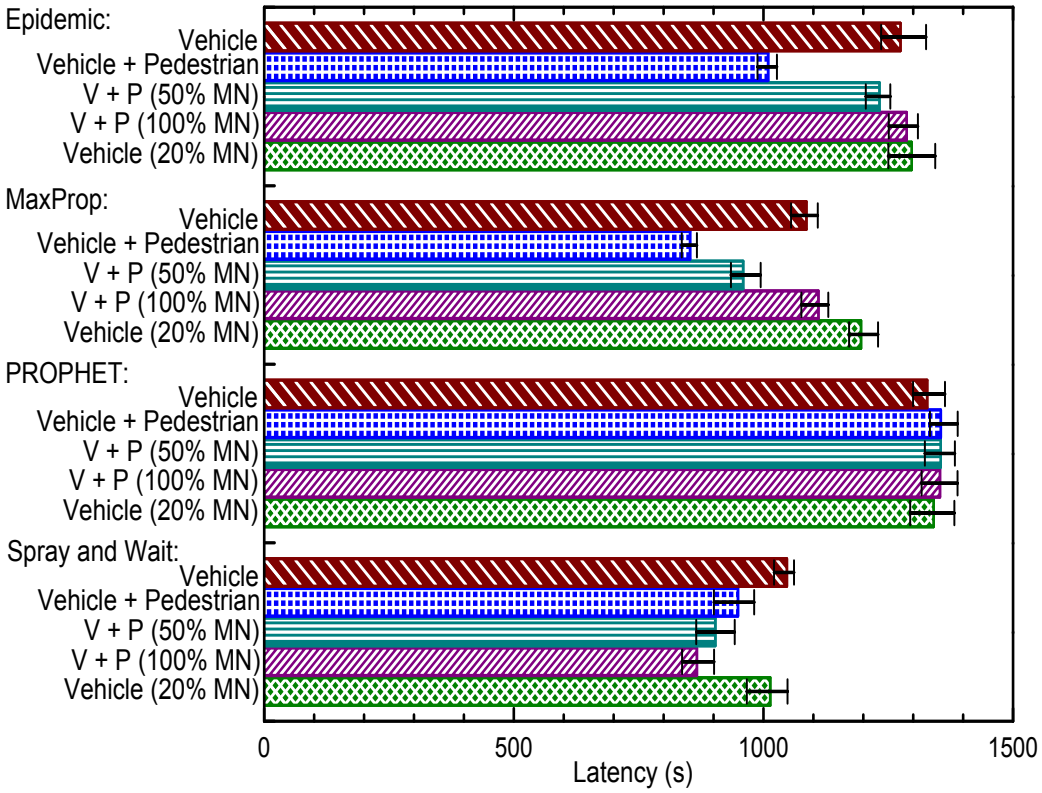


Figure 5.25: **Latency in the VDTN**

5.4.2.3 Overhead Rate in the VDTN

To better understand the system performance the max, min and average overhead rates are shown in Figure 5.26. The overhead rates define as how many hops does a bundle need to be successfully delivered to the destination (for the definition see section 3.4.2). When 280 benign pedestrians join in the network, the results obtained from Epidemic and MaxProp show that the overhead rates increase. Using Epidemic and MaxProp, the overhead rates also increase with 50% malicious pedestrian nodes in the system. Considering under the same conditions that the system obtains the higher delivery rates and shorter latency time, the increased overhead rates are acceptable. With 100% malicious pedestrian nodes, acting as blackhole attackers, the overhead rates decrease a bit. However, this does not imply higher performance: Blackhole attackers make the bundles have a high chance of getting lost; additionally, blackhole attackers have a negative effect on the delivery rates and latency time.

As we discussed in the delivery rate and latency parts, PROPHET does not perform well when random pedestrian nodes join the network, hence, the overhead rates increase a bit. The decreased overhead rates in Spray and Wait further confirm our

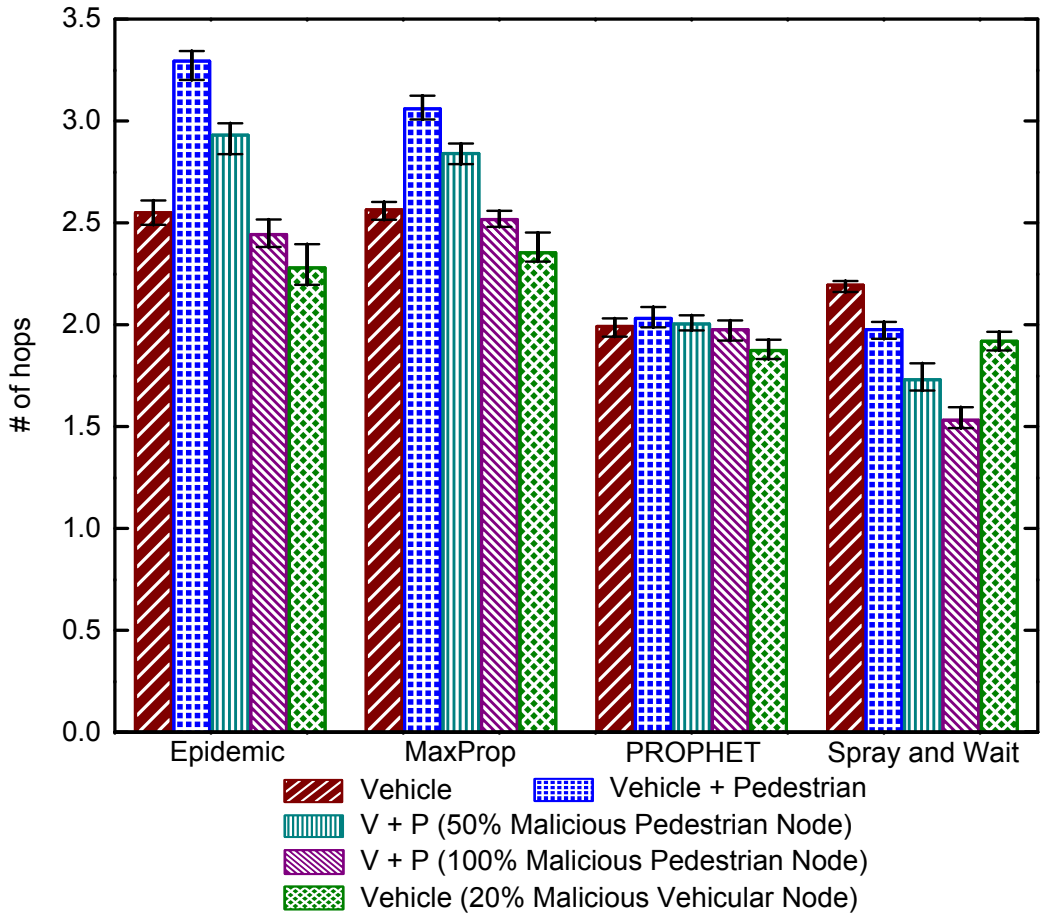


Figure 5.26: **Overhead Rate in the VDTN**

conclusions in the latency part. These successfully delivered bundles are the ones generated near the gateway as shown in Figure 5.26 of the decreased hops.

We also found the overhead rates decrease in all cases when there are 20% malicious vehicular nodes in the vehicular network. Keep in mind, that in the delivery rate and latency experiments, 20% malicious vehicular nodes affect the performance negatively. Hence, these decreased overhead rates mean that the malicious vehicular nodes block the chance for a bundle to be successfully transferred to the destinations.

5.4.3 Vehicles Supporting the Smartphone-based DTN

We also looked into the effect of a hybrid DTN from the perspective of the mobile users: Considering there is already a smartphone-based opportunistic network, where users can communicate with each other, the question is why they should cooperate

with vehicular nodes. As the environmental monitoring scenario is a pure machine-to-machine application there is no immediate benefit for mobile users acting as data mules as we assumed in the previous section. However, if we assume a TFT strategy where vehicles also transmit users' data, there might be enough incentive for mobile users to cooperate. Therefore, in this section we will evaluate whether the vehicles can improve the quality of service for applications running in the smartphone-based DTN.

As before our network consists of 28 vehicles and 280 pedestrians. In an interval between 15 and 25 seconds, a randomly chosen pedestrian generates one message for another pedestrian. That means in this experiment we have a real P2P communication pattern, compared to the sink-based network from the EMMA scenario in the previous section.

5.4.3.1 Delivery Rate and Latency in the Smartphone-based DTN

Figure 5.27 shows the delivery rates using four different routing protocols. When only the pedestrians exist in the mobile network, the average delivery rates are 63%, 64%, 23% and 15% for Epidemic, MaxProp, PROPHET and Spray and Wait respectively. In all scenarios, with the help from the vehicles, the delivery rates increase significantly, achieving 83%, 84%, 54% and 20% for the same routing protocols.

When looking at the latency in Figure 5.28 we see that even with the higher delivery rates the average latency decreases in all scenarios. When there are only the pedestrians in the mobile network, the average latency time is 1755 s, 1746 s, 1791 s and 1482 s for Epidemic, MaxProp, PROPHET and Spray and Wait respectively. In all scenarios, with the help from the vehicles, the latencies decrease significantly, achieving 1580 s, 1527 s, 1762 s and 1447 s for the same routing protocols.

These results show that the vehicles are able to shorten the latency of the successfully forwarded messages and lead to a better service quality.

5.4.3.2 Overhead in the Smartphone-based DTN

As seen from Figure 5.29, the overhead rates increase a bit in all cases. However, compared to the benefit from the delivery rates and the latency, these increased overhead rates do not have a strong negative effect on the performance of the system.

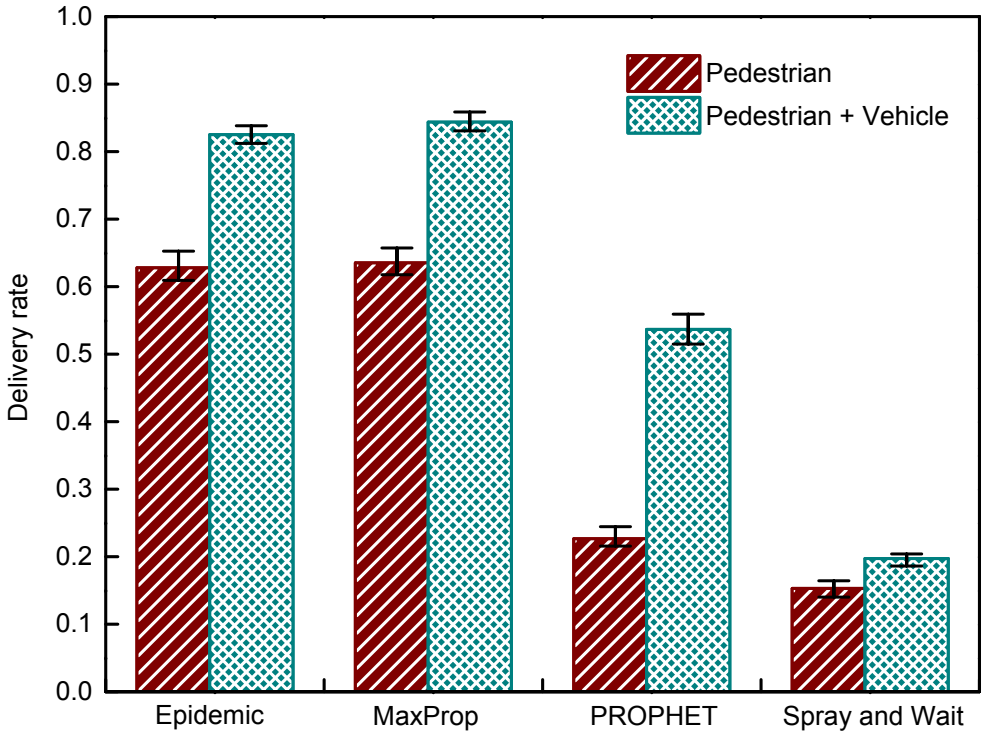


Figure 5.27: **Delivery Rate in the Smartphone-based DTN**

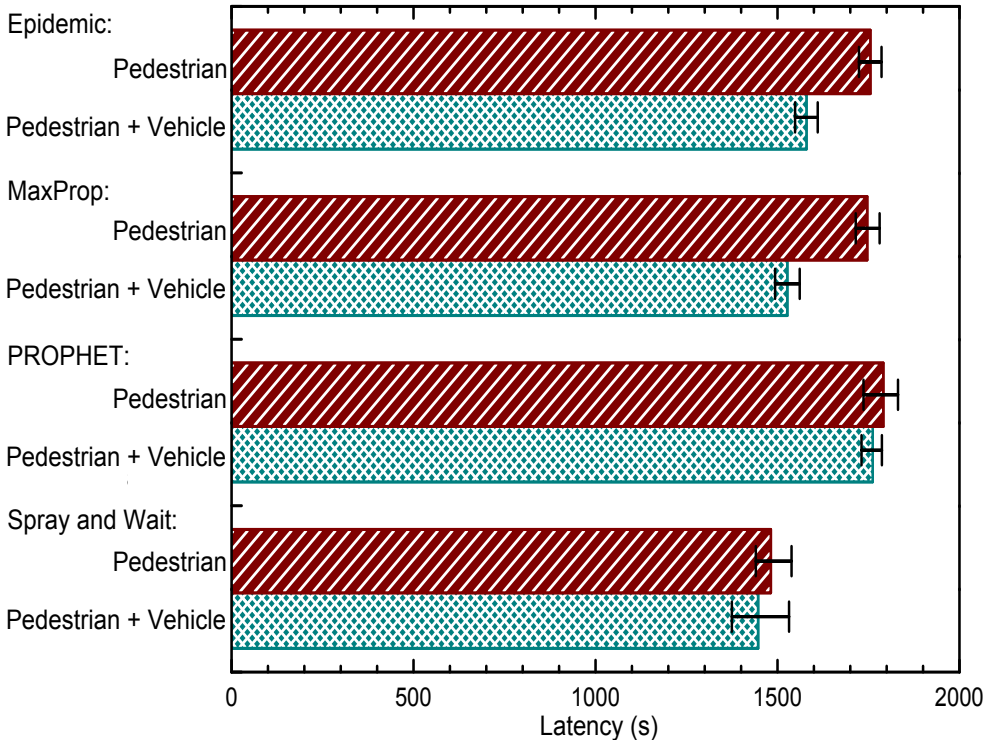


Figure 5.28: **Latency in the Smartphone-based DTN**

Due to the high movement speed, widely moving space and repeating behavioral patterns, vehicles can help the mobile network obtain better delivery rates, shorter latency time and acceptable overhead under different routing protocols. Overall these results show that for the pedestrians, the cooperation from the vehicles is a significant advantage, which is a strong incentive for them to also help the vehicles in return.

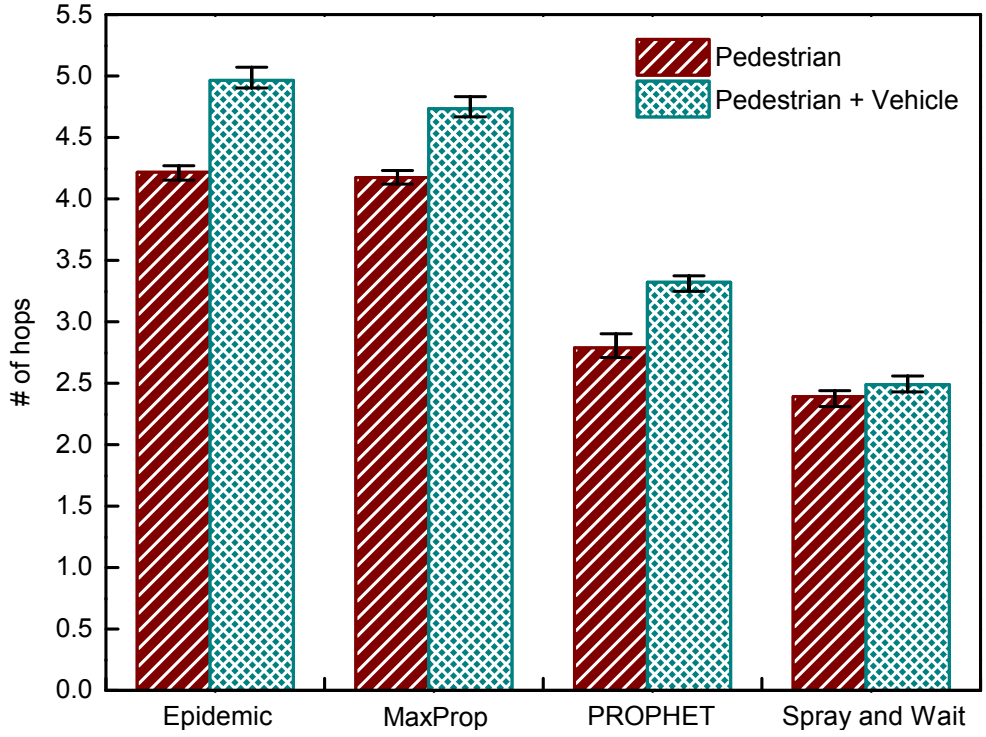


Figure 5.29: Overhead in the Smartphone-based DTN

5.4.4 MDS Performance

We look at the performance of the misbehavior detection in the network. As seen from the performance evaluations in section 5.4.2, it is crucial to detect misbehaving vehicles, while misbehaving pedestrians have only little impact on the system. As metric we use the average detection rates. We define the detection rate as the percentage of malicious nodes that have been detected by *all* good nodes as formula (3.6) in section 3.4.2. For a detection rate of 100%, we require that *all* malicious nodes are detected by *all* normal nodes.

From our previous part in section 5.3.2, we already know, our MDS has a good performance detecting malicious nodes in a purely vehicle-based network. The question we want to answer here is, whether the system can keep up this performance

with lots of pedestrian nodes exhibiting different characteristics compared to the vehicles in the network. The results of this test can be seen in Figure 5.30. For every routing protocol we performed a simulation with only the vehicles, and a second one where 280 pedestrians join the network. For each simulation we evaluated 3 situations where we randomly choose 3, 6 and 9 vehicles (10%, 20%, 30%) among the 28 vehicles as blackhole attackers.

It can be seen, that our MDS can achieve a detection rate of around 70% for all routing protocols when no pedestrian nodes are present in Figure 5.30. 70% is actually a very good result, as our definition of detection rate is very strict: A malicious node needs to be detected by *all* other nodes to count as "detected". However, in the Braunschweig system, some of the buses or trams never meet each other, hence, the detection rate cannot achieve 100%. The more often a node meets an attacker, the higher the chance it will detect the offender. So despite a lower detection rate according to the strict definition, the system is still immunized pretty well against offenders.

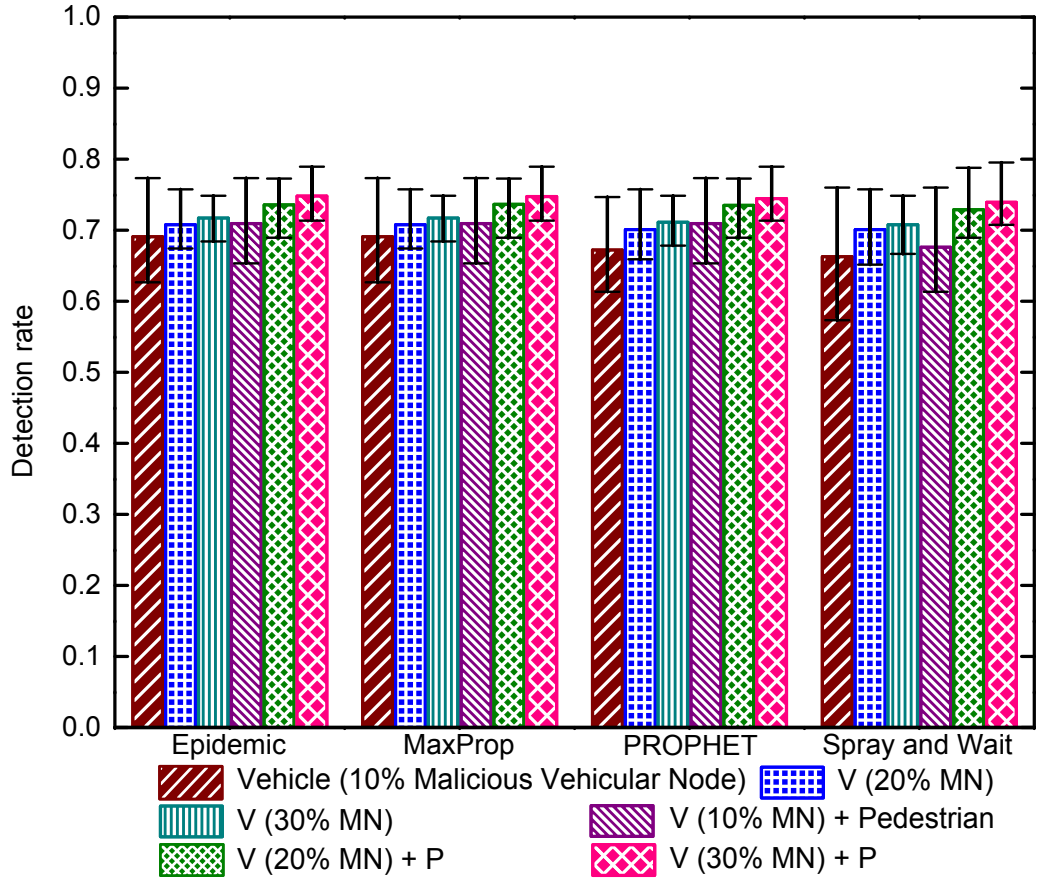


Figure 5.30: **Malicious Vehicle Detection Rate with and without Pedestrians**

In the second case we added the pedestrians which generate messages. To get comparable values, all pedestrians are benign, but we still choose 10%, 20% and 30% of the vehicles as blackhole attackers. The results of this scenario are also shown in Figure 5.30. We see that the MDS can sustain a detection rate of around 70% even though pedestrians participate in the network. Nodes with different patterns do not affect the ability of the MDS to discriminate between good and malicious behavior. This shows, that the clustering-based self-balancing MDS can not only adapt to different scenarios but can also deal with group heterogenous nodes with different movement and communication patterns.

While we have seen that malicious pedestrians can be tolerated, we want to check, whether they can be detected by the MDS. Generally this is a hard task, as an attacker can usually only be identified if several combined observations provide enough evidence to classify him as malicious. With a high number of slow moving pedestrians, some of them might only be contacted a few times or not at all. In Figure 5.31 we see the result of a scenario where 50% of the pedestrians are malicious. Again the amount of malicious vehicular nodes varies between 10% and 30%. Figure 5.31 presents two kind of detection results: How many malicious vehicles are detected by the vehicles ("Vehicle to Vehicle" in Figure 5.31) and how many malicious pedestrians are detected by the vehicles ("Vehicle to Pedestrian" in Figure 5.31). As in the previous results with benign pedestrians, the system can still obtain around 70% detection rate when vehicles detect evil vehicles. For Epidemic, MaxProp and PROPHET around 18% of the pedestrian attackers can be detected by vehicles and 11% for Spray and Wait.

These are good results. As evil vehicles have a more severe effect on the network performance they should be reliably detected. The MDS is able to do this no matter there are benign, malicious or no pedestrians at all. As expected, the detection rate of vehicles detecting pedestrians is not as high as the detection rate of vehicles detecting other vehicles. The reason is, that our MDS needs to learn and can only make a judgement relying on frequent encounters. However, vehicles do not meet all of the pedestrians. While the detection rate of vehicles detecting pedestrians is not so high, pedestrians more frequently in contact with the vehicles or even following vehicles actively trying to disrupt the network, have a higher chance of being detected by the MDS.

In fact, we think the small but significant pedestrian detection rate can have a good effect in a real system: Remember, that a smartphone-based DTN means, there are some users behind those phones who want to use the network service. Also, as we have seen in section 5.4.3, it is desirable for pedestrians to rely on vehicles to achieve a higher service quality. While the experiment shows that *most* of the time a malicious pedestrian is not detected, there is still a *significant* chance of being detected. The more frequently an attacker interacts with the system, the higher the

chance of detection. From a user's perspective the chance of being detected can be seen as similar to the chance of getting a speeding ticket: Most of the time you can exceed the speed limit without being detected. But there is still a non-negligible chance of being caught and punished, that provides enough incentive for most drivers not exceeding the speed limit too often. Similarly we think, the risk (up to 18%) of cheating the hybrid VDTN system for selfish reasons would be considered too high by most smartphone users, leading to less misbehaving nodes.

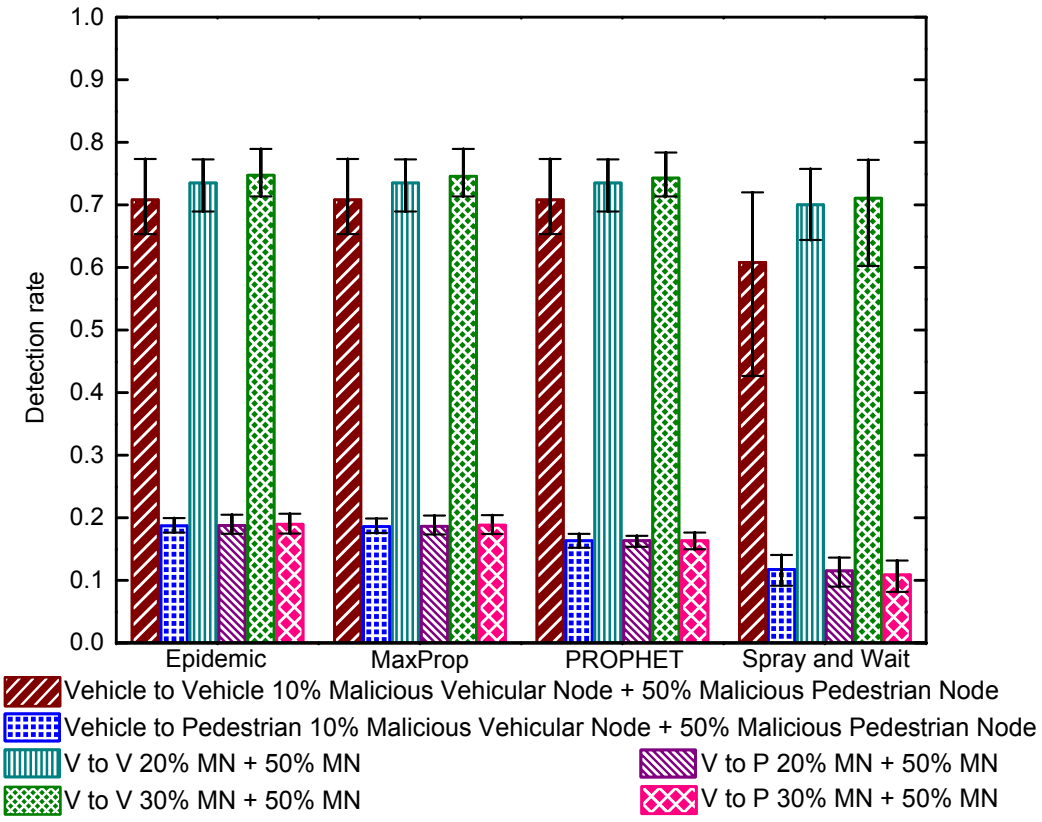


Figure 5.31: **Detection Rate in the Hybrid VDTN**

5.5 Conclusions

We presented a MDS that can detect malicious nodes not only in a dense vehicular network such as the presented Helsinki scenario but also in a wide and sparse real world scenario based on the San Francisco taxi GPS trace data and in a sink-based scenario which integrates an existing Braunschweig public transportation network to gather air pollution measurements. Compared to the previous work using threshold strategy the system only uses a few scenario-independent constants as parameters.

The wide difference between the evaluated scenarios is bridged by the dynamic cluster analysis, which can make the system perform well in different scenarios.

With independently operating nodes and asynchronous exchange of observations through encounter records, our MDS uses dynamic cluster analysis to exclude misbehaving nodes from the system. The evaluation results show, that our MDS is very well suited to protect the VDTNs, where there will be no continuous, ubiquitous network in the foreseeable future.

Later we presented a hybrid VDTN system, which consists of buses and trams running a machine-to-machine sensing application and pedestrians with private smartphones. We show, that both groups can profit from cooperation: Vehicles can obtain higher delivery rates and larger network capacities, especially when transferring bulk messages. Interestingly, this is independent of the fact that there might be large number of misbehaving mobile users. Mobile users get significantly better service quality due to the speed and covered area of the vehicles. While not the focus of this work, we have seen that not all routing protocols are equally suitable for this kind of the heterogeneous DTNs. Probably it might be a good idea to use different kind of cooperating routing protocols for the different parts of the network.

Our extensive simulations under different routing protocols demonstrate that in the hybrid VDTN, our MDS's learning and balancing mechanisms still work correctly when dealing with two groups of nodes with widely different behavior. While technically misbehaving mobile users cannot harm the system performance much, when a significant number of misbehaving users is detected by the MDS, it can provide a strong incentive for mobile users to cooperate.

Chapter 6

Conclusions

Recently vehicles are becoming increasingly intelligent, soon many will participate in a dynamic wireless network communicating with other vehicles nearby. Generally, communications between vehicles can enable various applications which can potentially provide specific and beneficial services for vehicles. However, to implement these applications, good connectivity is a key factor in vehicular networks. Delay- and Disruption-Tolerant Networking principles can guarantee the best networking performance in terms of delivery rate, reliability and latency even when dealing with intermittent connectivity.

However, such a system depends on cooperation. Malicious or selfish nodes may exist which can have a devastating effect on network performance. Faulty nodes can reduce the effectiveness of the system. Therefore, in this work we presented two cooperative MDSs to defend the network against the interference of faulty, selfish and malicious nodes. Our MDSs can adapt to different scenarios and different routing protocols. We presented simulation results showing that our approaches can not only efficiently detect evil nodes with a high detection rate and a low false positive rate but also can maintain a high delivery rate and low energy consumption.

6.1 Review of Contributions

This thesis makes the following contributions.

In Chapter 3 we described the basic framework of our MDS. The presented MDS uses encounter records as proof about nodes' behavior during previous contacts. On top of this information a reputation system is built that punishes bad behavior while encouraging cooperative behavior in the network. With independently operating nodes and asynchronous exchange of observations through encounter records, the

MDS is very well suited for protecting the security of VDTNs, where connectivity is intermittent and long delays are actually the norm.

In Chapter 4, we introduced a fixed threshold and an adaptive threshold MDS detecting blackhole and greyhole attackers. The MDS enables nodes in a VDTN to independently detect malicious nodes by distributing and combining information from previous encounters in the network. The system excludes malicious nodes from the network, and thus prevents them from further disrupting the network. The integrated reputation system encourages selfish nodes to cooperate. Extensive simulations under different routing protocols demonstrated that the adaptive threshold mechanism successfully takes the amount of information available to back a classification into account. This allows the system to achieve a high detection rate and a low false positive rate for different scenarios where the number of malicious nodes, the attack intensity or the employed routing protocol is varied. Furthermore, by exchanging classifications using the FBL mechanism, the MDS can boost detection efficiency.

In Chapter 5 we introduced a classifier based on dynamic cluster analysis, which enables the MDS to work without a long training phase or prior knowledge about the application. We evaluated how our MDS works in three homogeneous networks, a dense vehicular network in Helsinki, a wide and sparse real world scenario in San Francisco and an existing public transportation network in Braunschweig. Using encounter records documenting a node's behavior during previous contacts, our MDS uses dynamic cluster analysis to bridge the wide difference among the evaluated scenarios. This makes the system perform well in all three scenarios without the need to train and fine-tune the system to a specific scenario. We applied an improved TFT rule to update the trust reputation, that links incentive mechanisms with reputation. Furthermore, we introduced the FBL mechanism to help our MDS achieve a better network performance.

Moreover, we evaluated whether our MDS can still work efficiently when the nodes' behavior is heterogeneous. We first demonstrated that vehicular networks and mobile DTNs consisting of the evermore ubiquitous smartphones should be combined, as this can provide better service for both systems without compromising the integrity of either system. More importantly we have shown that our MDS can protect the security of this hybrid network. The evaluation results showed that our MDS is able to efficiently detect attackers and defend the hybrid network against the interference of malicious nodes.

6.2 Future Directions

Although our thesis presented a novel and efficient MDS to provide security for vehicular networks, there is always room for further improvement and extension of our MDS.

In Chapter 4, we used the idea of a threshold-based mechanism to build our MDS. The results showed that the threshold mechanism is suitable and reliable for closed networks with little changes. Relying on the number of ERs, we applied the adaptive threshold to cope with the problem that vehicular nodes have different degrees of participation in vehicular networks. It would be interesting to investigate how well our MDS can adapt to a wider range of scenarios regarding the node movement as well as network traffic patterns. It might be worthwhile looking into even more dynamic mechanisms where the MDS can adapt its rules while running. In such a case, the adaptive threshold will not only consider the degree of a node's participation but also the network's traffic patterns.

In Chapter 5, based on the *Re* sets in the ERs, we introduced the (*Message Forwarding Ratio* θ, *Message Receiving Ratio* ψ) tuples as input for the K-means clustering. It might be possible to derive further indicators for a node's behavior from the ERs (possibly by extending the ERs). Then the K-means clustering can be applied to a dataset of higher dimensionality which might improve the ability of the system to correctly discriminate between good and misbehaving nodes.

In our vehicular networks, we first considered the security of the system where homogenous cars freely move and organize themselves arbitrarily, forming a dynamic wireless topology. Furthermore, we evaluated the performance of the vehicular network where buses interact with pedestrians, forming a heterogenous network. With the increased popularity of mobile computing and communication devices, cars, buses, motorbikes, bicycles and pedestrians can all have the ability to communicate with other nearby nodes, composing an integrated and multipurpose vehicular network. In such a network, each node can act as an independent router and also as an application endpoint. It would be worthwhile to analyze, how the presented MDS mechanisms can be applied in such extremely hybrid networks, where nodes possess widely varying characteristics such as the movement or communication patterns or different hardware configurations. Considering different factors such as the transmission range, the buffer size and the movement speed, groups of cars, buses, motorbikes, bicycles and pedestrians have different characteristics therefore can provide varying levels of service to other nodes. Hence, a future MDS should judge nodes according to their abilities. Such a MDS could first divide nodes into different groups and then distinguish nodes' behavior compared to the nodes in the same group to ensure fair judgement of nodes with different abilities.

We evaluated our MDS to cope with a simple model of colluding attackers and obtained an efficient detection rate. However, in real applications even more advanced attack models are imaginable. Thus, evaluating and hardening the presented MDS approaches against new and sophisticated attack models seems a viable direction of research.

From a system's perspective, since the additional information exchanged via the FBL has proven to be very successful in this work, exchanging information via backbone networks when available, for example via roadside units or periodic cellular connections, might be able to improve the performance of the MDS even further.

Bibliography

[1] H. Moustafa and Y. Zhang. *Vehicular Networks: Techniques, Standards, and Applications*. Auerbach Publications, Boston, MA, USA, 1st edition, 2009.

[2] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-tolerant networking architecture. In *RFC 4838 (Informational)*, Apr. 2007.

[3] M. Raya and J.-P. Hubaux. The security of vehicular ad hoc networks. In *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks, SASN 2005*, pages 11–21, Alexandria, VA, USA, Nov. 2005.

[4] M. Raya, P. Papadimitratos, and J.-P. Hubaux. Securing vehicular communications. *Wireless Communications, IEEE*, 13(5):8–15, Oct. 2006.

[5] M. Raya, P. Papadimitratos, V.D. Gligor, and J.-P. Hubaux. On data-centric trust establishment in ephemeral ad hoc networks. In *Proceedings of the 27th Annual Joint Conference of the IEEE Computer and Communications, INFOCOM 2008*, pages 1238–1246, Apr. 2008.

[6] F. Kargl, P. Papadimitratos, L. Buttyan, M. Muter, E. Schoch, B. Wiedersheim, Ta-Vinh Thong, G. Calandriello, A. Held, A. Kung, and J.-P. Hubaux. Secure vehicular communication systems: implementation, performance, and research challenges. *Communications Magazine, IEEE*, 46(11):110–118, Nov. 2008.

[7] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology*, volume 196, pages 47–53, 1985.

[8] A. Seth and S. Keshav. Practical security for disconnected nodes. In *Proceedings of the 13th IEEE International Conference on Network Protocols, NPSec 2005*, pages 31–36, 2005.

[9] N. Asokan, Kari Kostiainen, Philip Ginzboorg, Jörg Ott, and Cheng Luo. Applicability of identity-based cryptography for disruption-tolerant networking. In *Proceedings of the 1st International MobiSys Workshop on Mobile Opportunistic Networking, MobiOpp 2007*, pages 52–56, Jun. 2007.

[10] A. Kate, G.M. Zaverucha, and U. Hengartner. Anonymity and security in delay tolerant networks. In *Proceedings of the 3rd International Conference on Security and Privacy in Communication Networks, SecureComm 2007*, pages 504–513, 2007.

[11] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, MobiCom 2000*, pages 255–265, Boston, USA, Aug. 2000.

[12] S. Buchegger and J.-Y. Le Boudec. Performance analysis of the confidant protocol. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2002*, pages 226–236, Lausanne, Switzerland, Jun. 2002.

[13] S. Buchegger and J.-Y. Le Boudee. Self-policing mobile ad hoc networks by reputation systems. *Communications Magazine, IEEE*, 43(7):101–107, Jul. 2005.

[14] P. Michiardi and R. Molva. Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Advanced Communications and Multimedia Security*, volume 100, pages 107–121. Springer US, 2002.

[15] F. Li, J. Wu, and A. Srinivasan. Thwarting blackhole attacks in disruption-tolerant networks using encounter tickets. In *Proceedings of the 28th Annual Joint Conference of the IEEE Computer and Communications, INFOCOM 2009*, pages 2428–2436, Rio de Janeiro, Brazil, Apr. 2009.

[16] Y. Ren, M.-C. Chuah, J. Yang, and Y. Chen. Detecting blackhole attacks in disruption-tolerant networks through packet exchange recording. In *Proceedings of IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks, WoWMoM 2010*, pages 1–6, 2010.

[17] Q. Li and G. Cao. Mitigating routing misbehavior in disruption tolerant networks. *Information Forensics and Security, IEEE Transactions on*, 7(2):664–675, Apr. 2012.

[18] I.F. Akyildiz, Ö.B. Akan, C. Chen, J. Fang, and W. Su. Interplanetary internet: state-of-the-art and research challenges. *Comput. Netw.*, 43(2):75–112, Oct. 2003.

[19] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. In *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems, ASPLOS 2002*, pages 96–107, San Jose, CA, USA, Oct. 2002.

[20] T. Small and Z.J. Haas. The shared wireless infostation model: A new ad hoc networking paradigm (or where there is a whale, there is a way). In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2003*, pages 233–244, Annapolis, Maryland, USA, 2003.

[21] A. Tovar, T. Friesen, K. Ferens, and B. McLeod. A dtn wireless sensor network for wildlife habitat monitoring. In *Proceedings of the 23rd Canadian Conference on Electrical and Computer Engineering, CCECE 2010*, pages 1–5, May 2010.

[22] K. Scott. Disruption tolerant networking proxies for on-the-move tactical networks. In *Proceedings of IEEE Military Communications Conference, MILCOM 2005*, volume 5, pages 3226–3231, Oct. 2005.

[23] R. Krishnan, P. Basu, J.M. Mikkelson, C. Small, R. Ramanathan, D.W. Brown, J.R. Burgess, A.L. Caro, M. Condell, N.C. Goffee, R. R. Hain, R.E. Hansen, C.E. Jones, V. Kawadia, D.P. Mankins, B.I. Schwartz, W.T. Strayer, J.W. Ward, D.P. Wiggins, and S.H. Polit. The spindle disruption-tolerant networking system. In *Proceedings of IEEE Military Communications Conference, MILCOM 2007*, pages 1–7, Orlando, Florida,USA, Oct. 2007.

[24] P. Holliday. Nomad a mobile ad hoc and distruption tolerant routing protocol for tactical military networks. In *Proceedings of the 29th IEEE International Conference on Distributed Computing Systems Workshops, ICDCS 2009*, pages 488–492, Jun. 2009.

[25] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. Delay-tolerant networking: an approach to interplanetary internet. *Communications Magazine, IEEE*, 41(6):128–136, Jun. 2003.

[26] W. Ivancic, W.M. Eddy, D. Stewart, L. Wood, P. Holliday, C. Jackson, and J. Northam. Experience with delay-tolerant networking from orbit. In *Proceedings of the 4th Advanced Satellite Mobile Systems Conference, ASMS 2008*, pages 173–178, Aug. 2008.

[27] C. Caini, P. Cornice, R. Firrincieli, and D. Lacamera. A dtn approach to satellite communications. *Selected Areas in Communications, IEEE Journal*, 26(5):820–827, Jun. 2008.

[28] C. Caini, P. Cornice, R. Firrincieli, M. Livini, and D. Lacamera. Tcp, pep and dtn performance on disruptive satellite channels. In *Proceedings of International Workshop on Satellite and Space Communications, IWSSC 2009*, pages 371–375, Sep. 2009.

[29] A. Doria, M. Uden, and D.P. Pandey. Providing connectivity to the saami nomadic community. In *Proceedings of the 2nd International Conference on Open Collaborative Design for Sustainable Innovation*, Dec. 2002.

[30] A. Lindgren and A. Doria. Experiences from deploying a real-life dtn system. In *Proceedings of IEEE Consumer Communications and Networking Conference, CCNC 2007*, pages 217–221, Jan. 2007.

[31] A.A. Hasson, R. Fletcher, and A. Pentland. Daknet: A road to universal broadband connectivity. In *Proceedings of the 2nd International Conference on Open Collaborative Design for Sustainable Innovation*, Bangalore, India, Dec. 2002.

[32] S. Guo, M. H. Falaki, E. A. Oliver, S. Ur Rahman, A. Seth, M. A. Zaharia, and S. Keshav. Very low-cost internet access using kiosknet. *SIGCOMM Comput. Commun. Rev.*, 37(5):95–100, 2007.

[33] J. Burgess, B. Gallagher, D. Jensen, and B.N. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications, INFOCOM 2006*, Barcelona, Spain, Apr. 2006.

[34] S. Lahde, M. Doering, W.-B. Poettner, G. Lammert, and L. Wolf. A practical analysis of communication characteristics for mobile and distributed pollution measurements on the road. *Wireless Communications and Mobile Computing*, 7(10):1209–1218, Dec. 2007.

[35] A. Pentland, R. Fletcher, and A. Hasson. Daknet: rethinking connectivity in developing nations. *Computer*, 37(1):78–83, Jan. 2004.

[36] S. Guo, M. Derakhshani, M.H. Falaki, U. Ismail, R. Luk, E.A. Oliver, S. Ur Rahman, A. Seth, M.A. Zaharia, and S. Keshav. Design and implementation of the kiosknet system. *Computer Networks*, 55(1):264–281, 2011.

[37] H. Soroush, N. Banerjee, M. Corner, B. Levine, and B. Lynn. A retrospective look at the umass dome mobile testbed. *SIGMOBILE Mob. Comput. Commun. Rev.*, 15(4):2–15, Mar. 2012.

[38] M. Doering, T. Pögel, and L. Wolf. Dtn routing in urban public transport systems. In *Proceedings of the 5th ACM workshop on Challenged networks, CHANTS 2010*, pages 55–62, Chicago, IL, USA, Sep. 2010.

[39] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. Wolf. IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation. *Electronic Communications of the EASST*, 37:1–11, Jan. 2011.

[40] H. Hartenstein and K.P. Laberteaux. A tutorial survey on vehicular ad hoc networks. *Communications Magazine, IEEE*, 46(6):164–171, Jun. 2008.

[41] P. Papadimitratos, A. La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza. Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation. *Communications Magazine, IEEE*, 47(11):84–95, Nov. 2009.

[42] Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. *Communications Surveys Tutorials, IEEE*, 8(1):24–37, Jan. 2006.

[43] L. Pelusi, A. Passarella, and M. Conti. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *Communications Magazine, IEEE*, 44(11):134–141, Nov. 2006.

[44] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, Duke University, 2000. http://issg.cs.duke.edu/pubs_old.html.

[45] J.A. Davis, A.H. Fagg, and B.N. Levine. Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks. In *Proceedings of the 5th International Symposium on Wearable Computers, ISWC 2001*, pages 141–148, 2001.

[46] K. Tan, Q. Zhang, and W. Zhu. Shortest path routing in partially connected ad hoc networks. In *Proceedings of IEEE Global Telecommunications Conference, GLOBECOM 2003*, volume 2, pages 1038–1042, Dec. 2003.

[47] T. Small and Z.J. Haas. Resource and performance tradeoffs in delay-tolerant wireless networks. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking, WDTN 2005*, pages 260–267, 2005.

[48] E.P.C. Jones, L. Li, J.K. Schmidtke, and P.A.S. Ward. Practical routing in delay-tolerant networks. *Mobile Computing, IEEE Transactions on*, 6(8):943–959, Aug. 2007.

[49] R. Ramanathan, R. Hansen, P. Basu, R. Rosales-Hain, and R. Krishnan. Prioritized epidemic for routing in opportunistic networks. In *Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking, Mobisys 2007*, pages 62–66, PR, USA, Jun. 2007.

[50] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, Jul. 2003.

[51] B. Burns, O. Brock, and B.N. Levine. Mora routing and capacity building in disruption-tolerant networks. *Ad Hoc Networks*, 6(4):600–620, 2008.

[52] A Lindgren and K.S. Phanse. Evaluation of queueing policies and forwarding strategies for routing in intermittently connected networks. In *Proceedings of the 1st International Conference on Communication System Software and Middleware, Comsware 2006*, pages 1–10, 2006.

[53] T.-K. Huang, C.-K. Lee, and L.-J. Chen. Prophet+: An adaptive prophet-based routing protocol for opportunistic network. In *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications, AINA 2010*, pages 112–119, Apr. 2010.

[54] F. McAtee, S. Narayanan, and G.G. Xie. Performance analysis of message prioritization in delay tolerant networks. In *Proceedings of IEEE Military Communications Conference, MILCOM 2012*, pages 1–6, Oct. 2012.

[55] T. Spyropoulos, K. Psounis, and C.S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259, Philadelphia, USA, Aug. 2005.

[56] T. Spyropoulos, K. Psounis, and C.S. Raghavendra. Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. In *Proceedings of the 5th Annual IEEE International Conference on Pervasive Computing and Communications, PerCom Workshops 2007*, pages 79–85, White Plains, New York, USA, Mar. 2007.

[57] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM 2004*, pages 145–158, Portland, Oregon, USA, 2004.

[58] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2004*, pages 187–198, Roppongi Hills, Tokyo, Japan, 2004.

[59] E.C.R. de Oliveira and C.V.N. de Albuquerque. Nectar: A dtn routing protocol based on neighborhood contact history. In *Proceedings of the 2009 ACM Symposium on Applied Computing, SAC 2009*, pages 40–46, Honolulu, Hawaii, 2009.

[60] T. Abdelkader, K. Naik, A. Nayak, and N. Goel. A socially-based routing protocol for delay tolerant networks. In *Proceedings of IEEE Global Telecommunications Conference, GLOBECOM 2010*, pages 1–5, Dec. 2010.

[61] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 1995.

[62] R. Housley and T. Polk. *Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2001.

[63] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Zhendong Ma, F. Kargl, A. Kung, and J.-P. Hubaux. Secure vehicular communication systems: design and architecture. *Communications Magazine, IEEE*, 46(11):100–109, Nov. 2008.

[64] P. Papadimitratos, G. Mezzour, and J.-P. Hubaux. Certificate revocation list distribution in vehicular communication systems. In *Proceedings of the 5th ACM international workshop on VehiculAr Inter-NETworking, VANET 2008*, pages 86–87, San Francisco, California, USA, 2008.

[65] A. Festag, G. Noecker, M. Strassberger, A. Lübke, B. Bochow, M. Torrent-moreno, S. Schnaufer, R. Eigner, C. Catrinescu, and J. Kunisch. Now - network on wheels: Project objectives, technology and achievements. In *Proceedings of the 5th International Workshop on Intelligent Transportation*, Hamburg, Deutschland, Mar. 2008.

[66] M. Torrent-Moreno. Inter-vehicle communications: assessing information dissemination under safety constraints. In *Proceedings of the 4th Annual Conference on Wireless on Demand Network Systems and Services, WONS 2007*, pages 59–64, Jan. 2007.

[67] P. Golle, D. Greene, and J. Staddon. Detecting and correcting malicious data in vanets. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks, VANET 2004*, pages 29–37, Philadelphia, PA, USA, Oct. 2004.

[68] M. Raya, P. Papadimitratos, I. Aad, D. Jungels, and J.-P. Hubaux. Eviction of misbehaving and faulty nodes in vehicular networks. *IEEE Journal on Selected Areas in Communications*, 25(8):1557–1568, Oct. 2007.

[69] R.K. Schmidt, T. Leinmüller, E. Schoch, A. Held, and G. Schäfer. Vehicle behavior analysis to enhance security in vanets. In *Proceedings of the 4th IEEE Vehicle-to-Vehicle Communications Workshop, V2VCOM 2008*, pages 1–8, 2008.

[70] R. Schmidt, T. Leinmüller, and A. Held. Defending against roadside attackers. In *Proceedings of the 16th World Congress on Intelligent Transport Systems, ITS 2009*, Sep. 2009.

[71] N. Bissmeyer, C. Stresing, and K.M. Bayarou. Intrusion detection in vanets through verification of vehicle movement data. In *Proceedings of the 2nd IEEE Vehicular Networking Conference, VNC 2010*, pages 166–173, Dec. 2010.

[72] M. Chuah and P. Yang. Comparison of two intrusion detection schemes for sparsely connected ad hoc networks. In *Proceedings of IEEE Military Communications Conference, MILCOM 2006*, pages 1–7, Washington DC, Oct. 2006.

[73] M. Chuah, P. Yang, and J. Han. A ferry-based intrusion detection scheme for sparsely connected ad hoc networks. In *Proceedings of the 4th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MobiQuitous 2007*, pages 1–8, Philadelphia, PA, USA, Aug. 2007.

[74] Y. Ren, M.C. Chuah, J. Yang, and Y. Chen. Muton: Detecting malicious nodes in disruption-tolerant networks. In *Proceedings of Wireless Communications and Networking Conference, WCNC 2010*, pages 1–6, Sydney, Australia, Apr. 2010.

[75] L. Buttyán and J.-P. Hubaux. Enforcing service availability in mobile ad-hoc wans. In *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking and computing, MobiHoc 2000*, pages 87–96, Boston, Massachusetts, USA, Aug. 2000.

[76] L. Buttyán and J.-P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *Mob. Netw. Appl.*, 8(5):579–592, Oct. 2003.

[77] H. Vogt, F.C. Gärtner, and H. Pagnia. Supporting fair exchange in mobile environments. *Mob. Netw. Appl.*, 8(2):127–136, Apr. 2003.

[78] S. Zhong, J. Chen, and Y.R. Yang. Sprite: a simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications, INFOCOM 2003*, volume 3, pages 1987–1997, Mar. 2003.

[79] R. Lu, X. Lin, H. Zhu, C. Zhang, P.-H. Ho, and X. Shen. A novel fair incentive protocol for mobile ad hoc networks. In *Proceedings of Wireless Communications and Networking Conference, WCNC 2008*, pages 3237–3242, 2008.

[80] A. Garyfalos and K.C. Almeroth. Coupons: A multilevel incentive scheme for information dissemination in mobile networks. *Mobile Computing, IEEE Transactions on*, 7(6):792–804, 2008.

[81] S.-B. Lee, G. Pan, J.-S. Park, M. Gerla, and S. Lu. Secure incentives for commercial ad dissemination in vehicular networks. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing, MobiHoc 2007*, pages 150–159. ACM, 2007.

[82] H. Zhu, X. Lin, R. Lu, Y. Fan, and X. Shen. Smart: A secure multilayer credit-based incentive scheme for delay-tolerant networks. *Vehicular Technology, IEEE Transactions on*, 58(8):4628–4639, 2009.

[83] R. Lu, X. Lin, H. Zhu, X. Shen, and B. Preiss. Pi: A practical incentive protocol for delay tolerant networks. *Wireless Communications, IEEE Transactions on*, 9(4):1483–1493, Apr. 2010.

[84] R. Axelrod and W.D. Hamilton. The evolution of cooperation. *Science*, 211(4489):1390–1396, 1981.

[85] B. Cohen. Incentives build robustness in bittorrent. In *Proceedings of Workshop on Economics of Peer-to-Peer systems*, 2003.

[86] D. Hales. From selfish nodes to cooperative networks - emergent link-based incentives in peer-to-peer networks. In *Proceedings of the 4th International Conference on Peer-to-Peer Computing 2004*, pages 151–158, 2004.

[87] G. Neglia, G. Presti, H. Zhang, and D. Towsley. A network formation game approach to study bittorrent tit-for-tat. In *Network Control and Optimization*, volume 4465 of *Lecture Notes in Computer Science*, pages 13–22. 2007.

[88] V. Srinivasan, P. Nuggehalli, C. F Chiasserini, and R.R. Rao. Cooperation in wireless ad hoc networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computerand Communications, INFOCOM 2003*, volume 2, pages 808–817, 2003.

[89] F. Milan, J.J. Jaramillo, and R. Srikant. Achieving cooperation in multihop wireless networks of selfish nodes. In *Proceeding of the 2006 workshop on Game theory for communications and networks, GameNets 2006*, 2006.

[90] J.J. Jaramillo and R. Srikant. Darwin: distributed and adaptive reputation mechanism for wireless ad-hoc networks. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking, MobiCom 2007*, pages 87–98, New York, NY, USA, 2007.

[91] U. Shevade, H.H. Song, L. Qiu, and Y. Zhang. Incentive-aware routing in dtns. In *Proceedings of IEEE International Conference on Network Protocols, ICNP 2008*, pages 238–247, Oct. 2008.

[92] U. Lee, B. Zhou, M. Gerla, E. Magistretti, P. Bellavista, and A. Corradi. Mobeyes: smart mobs for urban monitoring with a vehicular sensor network. *Wireless Communications, IEEE*, 13(5):52–57, 2006.

[93] H. Zhu, R. Lu, X. Shen, and X. Lin. Security in service-oriented vehicular networks. *Wireless Communications, IEEE*, 16(4):16–22, 2009.

[94] F. Li and J. Wu. Frame: An innovative incentive scheme in vehicular networks. In *Proceedings of IEEE International Conference on Communications, ICC 2009*, pages 1–6, 2009.

[95] C. Hernández-Goya, P. Caballero-Gil, J. Molina-Gil, and C. Caballero-Gil. Cooperation enforcement schemes in vehicular ad-hoc networks. In Roberto Moreno-Díaz, Franz Pichler, and Alexis Quesada-Arencibia, editors, *Computer Aided Systems Theory - EUROCAST 2009*, pages 429–436. Springer-Verlag, Berlin, Heidelberg, 2009.

[96] S.-B. Lee, J.-S. Park, M. Gerla, and S. Lu. Secure incentives for commercial ad dissemination in vehicular networks. *Vehicular Technology, IEEE Transactions on*, 61(6):2715–2728, Jul. 2012.

[97] P. Papadimitratos, L. Buttyan, J-P Hubaux, F. Kargl, A. Kung, and M. Raya. Architecture for secure and private vehicular communications. In *Proceedings of the 7th International Conference on ITS Telecommunications, ITST 2007*, pages 1–6, 2007.

[98] L. TamilSelvan and V. Sankaranarayanan. Prevention of blackhole attack in manet. In *Proceedings of the 2nd International Conference on Wireless Broadband and Ultra Wideband Communications, AusWireless 2007*, pages 21–21, 2007.

[99] J. Hortelano, J.C. Ruiz, and P. Manzoni. Evaluating the usefulness of watchdogs for intrusion detection in vanets. In *Proceedings of IEEE International Conference on Communications Workshops, ICC 2010*, pages 1–5, 2010.

[100] G. Gupta, P. Nagrath, S. Aneja, and N. Gupta. Reference based approach to mitigate blackhole attacks in delay tolerant networks. In *Proceedings of the 8h ACM symposium on QoS and security for wireless and mobile networks, Q2SWinet 2012*, pages 85–88, 2012.

[101] H.L. Nguyen and U.T. Nguyen. A study of different types of attacks in mobile ad hoc networks. In *Proceedings of the 25th IEEE Canadian Conference on Electrical Computer Engineering, CCECE 2012*, pages 1–6, Apr. 2012.

[102] R. Lu, X. Lin, X. Liang, and X. Shen. Sacrificing the plum tree for the peach tree: A socialspot tactic for protecting receiver-location privacy in vanet. In *Proceedings of IEEE Global Telecommunications Conference, GLOBECOM 2010*, pages 1–5, 2010.

[103] Z. Gao, H. Zhu, S. Du, C. Xiao, and R. Lu. Pmds: A probabilistic misbehavior detection scheme in dtn. In *Proceedings of IEEE International Conference on Communications, ICC 2012*, pages 4970–4974, 2012.

[104] H. Zhu, S. Du, Z. Gao, M. Dong, and Z. Cao. A probabilistic misbehavior detection scheme towards efficient trust establishment in delay-tolerant networks. *Parallel and Distributed Systems, IEEE Transactions on*, (99):1, 2013.

[105] A. Keränen, J. Ott, and T. Kärkkäinen. The one simulator for dtn protocol evaluation. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques, Simutools 2009*, pages 55:1–55:10, Rome, Italy, Mar. 2009.

[106] A. Keränen, T. Kärkkäinen, and J. Ott. Simulating mobility and dtns with the one (invited paper). *Journal of Communications*, 5(2), 2010.

[107] G. Sandulescu and S. Nadjm-Tehrani. Opportunistic dtn routing with window-aware adaptive replication. In *Proceedings of the 4th Asian Conference on Internet Engineering, AINTEC 2008*, pages 103–112, 2008.

[108] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser. CRAWDAD data set epfl/mobility (v. 2009-02-24). http://crawdad.cs.dartmouth.edu/epfl/mobility, Feb. 2009.

[109] Y. Guo, S. Schildt, J. Morgenroth, and L. Wolf. A misbehavior detection system for vehicular delay tolerant networks. In *Proceedings of the INFORMATIK 2012*, Braunschweig, Germany, Sep. 2012.

[110] Y. Guo, S. Schildt, and L. Wolf. Detecting blackhole and greyhole attacks in vehicular delay tolerant networks. In *Proceedings of the 5th International Conference on Communication Systems and Networks, COMSNETS 2013*, pages 1–7, Bangalore, India, Jan. 2013.

[111] Y. Guo, S. Schildt, T. Pögel, and L. C. Wolf. Detecting malicious behavior in a vehicular DTN for public transportation. In *Proceedings of Global Information Infrastructure and Networking Symposium , GIIS 2013*, pages 1–8, Trento, Italy, Oct. 2013.

[112] Y. Guo, S. Schildt, and L. Wolf. Using cluster analysis to detect attackers in vehicular delay tolerant networks. In Mostafa Hashem Sherif, Abdelhamid Mellouk, Jun Li, and Paolo Bellavista, editors, *Ad Hoc Networks*, volume 129 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 181–196. Springer International Publishing, 2014.

[113] J. Morgenroth, S. Schildt, and L. Wolf. A bundle protocol implementation for Android devices. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, Mobicom 2012*, page 443, New York, USA, Aug. 2012.

[114] Y. Guo, S. Schildt, T. Pögel, S.Rottmann, and L. Wolf. Mitigating blackhole attacks in a hybrid VDTN. In *Proceedings of the 1st International Workshop on Smart Vehicles: Connectivity Technologies and ITS Applications, SmartVehicles 2014*, Sydney, Australia, Jun. 2014.

[115] M.R. Anderberg. *Cluster analysis for applications*. Academic Press, 1973.

[116] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, Mar. 1990.

[117] A.K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).

[118] D.J. Ketchen and C.L. Shook. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic Management Journal*, 17(6):441–458, Jun. 1998.

[119] A.K. Jain and R.C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[120] A. Dharmarajan and T. Velmurugan. Applications of partition based clustering algorithms: A survey. In *Proceedings of IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2013*, pages 1–5, Dec. 2013.

[121] M.A.T. Figueiredo and A.K. Jain. Unsupervised learning of finite mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):381–396, 2002.

[122] M.H. Hansen and B. Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774, 2001.

[123] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.